



MINISTÉRIO DA CIÊNCIA E TECNOLOGIA

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-14131-TDI/1081

**UM SERVIÇO DE COORDENAÇÃO DE PROCESSOS
INTEGRADO AO AMBIENTE DE ENGENHARIA DE SOFTWARE
E-WEBPROJECT**

Moacyr Gonçalves Cereja Junior

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada,
orientada pelo Dr. Nilson Sant'Anna, aprovada em
30 de março de 2004.

INPE
São José dos Campos
2006

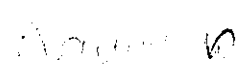
681.3.06

Cereja Junior, M. G.

Um serviço de coordenação de processos integrado ao ambiente de engenharia de software e-webproject / M. G. Cereja Junior. – São José dos Campos: INPE, 2004.
274 p. ; (INPE-14131-TDI/1081).

1. Processos de software. 2. Modelagem de processos.
3. Automação de processos. 4. Engenharia de software.
5. Agentes. I.Título.

Dr. Solon Venâncio de Carvalho



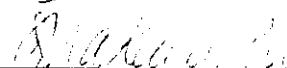
Presidente/INPE, SJCampos-SP

Dr. Nilson Sant'Anna



Orientador/INPE, SJCampos-SP

Dr. Tatuó Nakanishi



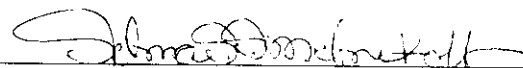
Membro da Banca/ INPE, SJCampos-SP

Dr. Maurício Gonçalves Vieira Ferreira



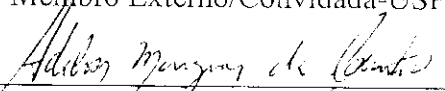
Membro da Banca/ INPE, SJCampos-SP

Dra. Selma Shin Shimizu Melnikoff



Membro Externo/Convidada-USP

Dr. Adilson Marques da Cunha



Membro Externo/Convidado-ITA

Aluno: Moacyr Gonçalves Cereja Júnior

São José dos Campos, 30 de março de 2004.

“Nem tudo o que dá certo é certo”

David Capistrano

*Dedico este trabalho a minha mãe Deocília
E a memória de meu pai Moacyr.*

AGRADECIMENTOS

Agradeço a Deus, por sua grandeza, por me dar a oportunidade de fazer este trabalho já que existem muitos, mais competentes e inteligentes do que eu, que talvez, não puderam ter a mesma oportunidade. A quem procuro nos momentos mais difíceis e angustiantes, que me dá forças para superar os obstáculos.

Agradeço ao meu orientador e amigo Nilson Sant'Anna que me abriu portas desde a época em que eu era seu aluno de graduação. Que me auxiliou para que eu chegasse e concluísse esta etapa de minha vida, contribuindo com valiosas doses de conhecimento, oportunidades e trabalho.

Agradeço a meus irmãos: Isabel, Eduardo, Renata e Rafael, pelo apoio que me deram durante o desenvolvimento deste trabalho. E a meus parentes: tios, tias, primos e primas que de alguma forma contribuíram para que eu chegasse até aqui.

Agradeço a todos os professores do INPE, que não medem esforços para dar uma formação ímpar aos alunos da CAP, não só com referência às disciplinas que ministram, mas também não negam informações quando solicitados, a qualquer momento, mesmo estando, em muitas vezes, atarefados.

Agradeço a todos que trabalham ou trabalharam na SESIS – Sistemas de Engenharia de Software que me auxiliaram a concluir este trabalho.

Por último, agradeço aos meus amigos pessoais em especial as figuras: Luiz Fernando (Horse), Elias, Marcio, Anderson (Mirinda), Anderson (Magrão), Genilson e Adilson; meus amigos do INPE; e ao pessoal da república Órion 147: Elcio, Felipe, Jeferson e Joanito, pela bagunça, piadas e outras besteiras que nos divertem, fazendo-nos esquecer os problemas, quando as coisas estão ficando pretas, renovando o bom humor para continuar o trabalho.

RESUMO

Organizações que desenvolvem software, para ganhar competitividade, estão em escala crescente, se preocupando com modelos de qualidade como o CMM, CMMI, e SPICE. Estes modelos possuem, como elemento central de suas especificações, o conceito de “processos”. Ambientes de Engenharia de Software se apresentam como uma solução tecnológica ao efetivo apoio a esforços de melhoria de processos. Este trabalho apresenta as características, modelagem e um protótipo inicial de um serviço de coordenação de processos de software integrado ao ambiente de Engenharia de Software e-WebProject, capaz de apoiar todo o ciclo de vida de um processo, desde a sua definição até a sua execução.

A SOFTWARE PROCESS COORDINATION SERVICE INTEGRATED INTO THE E-WEBPROJECT PROCESS-CENTERED SOFTWARE ENGINEERING ENVIRONMENT

ABSTRACT

Organizations that develop Software, to have more competitiveness, are increasing the use of maturity models like CMM, CMMI and SPICE. These models have as specification main element, the “process” concept. Process-Centered Software Engineering Environments (PSEEs) are presented as a technological solution to software process improvement practices. This work presents features, a modeling and an initial prototype for a software process coordination service integrated into the e-WebProject PSEE, able to support all the software process life cycle: from software process definition to software process enaction.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

CAPÍTULO 1 - INTRODUÇÃO	29
1.1. - Motivação	32
1.2. - Objetivo do trabalho de pesquisa	35
1.3. - Estrutura do Trabalho	36
CAPÍTULO 2 - PROCESSOS DE SOFTWARE E MODELAGEM DE PROCESSOS	39
2.1. - Definições	41
2.2. - Porque processos de software são importantes	42
2.3. - Modelagem de Processos de Software	48
2.3.1. - Definição de Modelo de Processo	48
2.3.2. Razões para se modelar um Processo	48
2.3.3. – Histórico	51
2.4. - Linguagens de Modelagem de Processo (Process Modeling Languages – PMLs)	53
2.4.1. Elementos de Processo	54
2.4.2. - Fases de modelagem e automação de um processo	57
CAPÍTULO 3 – AMBIENTES DE ENGENHARIA DE SOFTWARE CENTRADOS EM PROCESSO E AUTOMAÇÃO DE PROCESSOS	63
3.1. - Ambientes de Engenharia de Software Centrados em Processo (Process- Centered Software Engineering Environments – PSEE)	63
3.2. - O Ambiente e-WebProject®	73
3.2.1. Requisitos Definidos para o e-WebProject	74
3.2.2. Arquitetura Proposta	75
3.2.3. O apoio à execução de processos no e-WebProject	77

3.2.3.1. O Processo: Ciclo de Vida	77
3.2.3.2. - A fase de planejamento de um processo	78
3.2.3.3. - A fase de execução e acompanhamento do processo	81
3.2.3.4. - A fase de avaliação e melhoria do processo	83
3.3. - Tecnologia de Workflow	84
3.3.1. - Definições de fluxo de trabalho (Workflow)	84
3.3.2. - Sistemas de Gerenciamento de Workflow	85
3.3.3. - O Workflow Management Coalition (WFMC)	89
3.3.4. Um modelo de referência para implementação de sistemas de gerenciamento de workflow	89
3.4. - Tecnologia de Agentes	91
3.4.1. - Definição de Agente	91
3.4.2. - Tipologia de Agentes	93
3.4.3. - Uma Visão Panorâmica dos Diferentes Tipos de Agentes	96
3.4.3.1. - Agentes Colaborativos	96
3.4.3.2. - Agentes de Interface	97
3.4.3.3. - Agentes de Software Reativos	99
3.4.3.4. - Agentes Inteligentes	99
3.4.4. - Perspectivas tecnológicas para implementação de sistemas multi- agentes	100
3.4.4.1. - Especificação FIPA Abstract Architecture	100
3.4.4.2. - A plataforma de Agentes JADE	102

CAPÍTULO 4 – O SERVIÇO DE COORDENAÇÃO DE PROCESSOS DE SOFTWARE105

4.1. - Escopo do Serviço de Coordenação de Processos	105
4.2. - Requisitos para o Serviço de Coordenação de Processos	106
4.3. - Uma visão geral do serviço de coordenação de processos	107
4.4. - O Ambiente de Definição de Processos	111
4.4.1. - Os Componentes do Ambiente	111
4.5. - O Repositório de Artefatos	113

4.6. - A Máquina de Processos	115
4.7. - A Arquitetura Física	117
4.7.1. - Descrição dos Componentes da Arquitetura de Desenvolvimento	119

CAPÍTULO 5 – O AMBIENTE DE DEFINIÇÃO DE PROCESSOS DE SOFTWARE123

5.1. Modelo Descritivo	123
5.1.1. - Domínio do Negócio.	123
5.1.2. - Descrição do Negócio.....	123
5.2. - Modelo Conceitual.....	125
5.2.1. - Objetivo do Sistema	125
5.2.2. - Atores e Funcionalidades.....	125
5.2.3. - Visão Use Case: Contexto do Sistema.....	126
5.2.4. - Descrição dos Casos de Uso	127
5.2.4.1. Caso de Uso: Manter Modelos de Processo.....	127
5.3. - Modelo de Negócios.....	128
5.3.1. Modelo de Classes	128
5.3.2. Modelagem da Interação entre os Objetos.....	130
5.3.2.1. - A Seqüência: Caso de Uso Manter Modelos de Processo	130

CAPÍTULO 6 – A MÁQUINA DE PROCESSOS133

6.1. - Modelo Descritivo	133
6.1.1. - Domínio do Negócio.....	133
6.1.2. - Descrição do Negócio.....	133
6.2. - Modelo Conceitual.....	135
6.2.1. - Objetivo do Sistema	135
6.2.2. - Atores & Funcionalidades	135
6.2.3. - Visão Use Case: Contexto do Sistema.....	136
6.2.4. Descrição dos Agentes	139
6.2.5. - Descrição dos Casos de Uso	139
6.2.5.1. - Caso de Uso: Controlar Instâncias de Processos (Coordenar execução).....	140

6.3. - Modelo de Negócios.....	141
6.3.1. - Modelagem Estrutural.....	141
6.3.1.1. - Modelo de Classes Persistentes	141
6.3.1.2. - Restrições para ativação e conclusão de tarefas	143
6.3.1.3. - Modelo de Classes do Serviço de Agentes	146
6.3.2. - Modelagem Comportamental.....	147
6.3.2.1. - Os estados de um processo.....	147
6.3.2.2. - Os estados de uma tarefa	148
6.3.2.3. - O protocolo de execução de processos	150
6.3.2.4. - Modelagem da comunicação entre agentes.....	152
6.3.2.5. - Modelagem do processamento interno dos agentes.	155
6.3.2.5.1. - O comportamento interno do agente instanciador	155
6.3.2.5.2. - O comportamento interno do agente coordenador	157
6.3.2.5.3. - O comportamento interno do agente planejador	158
6.3.2.5.4. - O comportamento interno do agente mensageiro.....	159
6.3.2.5.5. - O comportamento interno do agente monitor	160
CAPÍTULO 7 – O REPOSITÓRIO DE ARTEFATOS	163
7.1. - Modelo Descritivo	163
7.1.1. - Domínio do Negócio.....	163
7.1.2. - Descrição do Negócio.....	163
7.2. - Modelo Conceitual.....	165
7.2.1. - Objetivo do Sistema	165
7.2.2. - Atores & Funcionalidades	165
7.2.3. - Visão Use Case: Contexto do Sistema.....	167
7.2.4. - Descrição dos Casos de Uso	167
7.2.4.1. - Caso de Uso: Manter Artefatos.....	168
7.3. - Modelo de Negócios.....	169
7.3.1. - Modelagem Estrutural.....	169
7.3.1.1. - Modelo de Classes Persistentes	169
CAPÍTULO 8 – UM PROTÓTIPO DO SERVIÇO	173

8.1.- Definição do Processo	173
8.1.1. – Propósito.....	174
8.1.2. - Atividades do Processo	174
8.2. - Modelagem do Processo	175
8.3. - Configuração do Modelo de Processo no e-WebProject	176
8.4. - Configuração de uma instância de modelo de processo no e- WebProject.....	179
8.5. Alocação de uma tarefa de processo para um executor no ambiente	179
CAPÍTULO 9 – CONSIDERAÇÕES FINAIS.....	181
9.1. – Conclusões.....	183
9.2. - Contribuições.....	183
9.3. - Perspectivas de trabalhos futuros.....	184
REFERÊNCIAS BIBLIOGRAFICAS	187
APÊNDICE A – ALGUMAS ABORDAGENS PARA A MODELAGEM DE PROCESSOS	197
APÊNDICE B – CÓDIGO FONTE DO AGENTE MENSAGEIRO.....	227
APÊNDICE C – DESCRIÇÃO DOS CASOS DE USO E DIAGRAMAS DE SEQÜENCIA DO AMBIENTE DE DEFINIÇÃO DE PROCESSOS	231
APÊNDICE D – DESCRIÇÃO DOS CASOS DE USO DA MÁQUINA DE PROCESSOS	259
APÊNDICE E- DESCRIÇÃO DOS CASOS DE USO DO REPOSITÓRIO DE ARTEFATOS	265

LISTA DE FIGURAS

2.1.- O processo como uma caixa preta (1).	43
2.2.- O processo como uma caixa preta (2).	44
2.3.- Um processo transparente.....	47
3.1.- O PSEE centraliza e integra o apoio para engenharia do processo, gerência de projetos e engenharia de software.....	65
3.2.- Componentes e camada conceitual do ambiente.	76
3.3.- Arquitetura física do ambiente	77
3.4.- O Caráter evolutivo do Ambiente	77
3.5.- Ciclo de vida de um processo.....	79
3.6.- A fase de planejamento	80
3.7.- Os estados possíveis para um processo quando na fase de execução.	82
3.8.- Características básicas de um sistema de gerenciamento de workflow.	88
3.9.- Estrutura genérica de um produto de workflow.	90
3.10.- Os agentes percebem as alterações de seu ambiente de agem de a modo a modificá-los.	92
3.11.- Uma classificação de agentes de software.....	94
3.12.- Uma visão da tipologia de agentes.....	95
3.13.- Comunicação de uma colônia de 5 agentes.....	97
3.14.- Como Agentes de Interface Trabalham.	98
3.15.- Arquitetura abstrata FIPA mapeada para várias implementações concretas.....	102
4.1.- Os componentes do Serviço de Coordenação de Processos	109
4.2.- A interação entre o serviço de coordenação de processos e seus participantes..	111
4.3.- Base conceitual da ferramenta PDE.....	112
4.4.- Componentes da Arquitetura	118
5.1.- Atividades do Domínio Modelar Processos	124
5.2.- Diagrama de Casos de Uso: Ambiente de Definição de Processos	126
5.3.- Modelo de classes persistentes: repositório de modelos de processos.....	129

5.4.- Diagrama de seqüência: inserir um novo modelo de processo	131
6.1.- Atividades do negócio executar processos	135
6.2.- Diagrama de Contexto: Máquina de Processos	137
6.3.- Diagrama de Casos de Uso: sub-módulos da máquina de processos	138
6.4.- Diagrama de classes persistentes: repositório de instâncias de processos	142
6.5.- Diagrama de classes: serviço de Agentes.....	146
6.6.- Estados de uma instância de processo.....	148
6.7.- Estados de uma tarefa	149
6.8.- Diagrama de atividades que apresenta o protocolo de execução de processos entre os agentes.	152
6.9.- Diagrama de colaboração: interação entre os agentes	153
6.10.- Diagrama de seqüência: troca de mensagens dos agentes	154
6.11.- Diagrama de atividades mostrando o comportamento interno do agente instanciador.....	156
6.12.- Diagrama de atividades mostrando o comportamento interno do agente coordenador.	157
6.13.- Diagrama de atividades mostrando o comportamento interno do agente planejador....	159
6.14.- Diagrama de atividades mostrando o comportamento interno do agente messaging.	160
6.15.- Diagrama de atividades mostrando o comportamento interno do agente monitor.	161
7.1.- Atividades do negócio controlar artefatos.....	165
7.2.- Diagrama de Casos de Uso: Repositório de Artefatos	167
7.3.- Diagrama de classes: repositório de artefatos.....	170
8.1.- Modelo de Processo do Processo de Solução de Problemas	175
8.2.- Navegador de Modelos de Processos da ferramenta PDE	176
8.3.- Propriedades de uma tarefa do Processo.	177
8.4.- Configuração de papéis responsáveis de uma tarefa	177
8.5.- Configuração de artefatos de entrada de uma tarefa	178
8.6.- Configuração de artefatos de saída de um processo.....	178

8.7.- Configuração de um instância de modelo de processo.....	179
8.8.- Espaço de trabalho de um usuário do e-WebProject	180

LISTA DE TABELAS

1.1.- Custos e ganhos de 13 organizações que adotaram práticas de esforços de melhorias de processos baseados no modelo CMM.....	33
1.2.- Experiências organizacionais ilustrando os custos e benefícios da adoção de programas de SPI.	34
5.1.- Classes do repositório de modelos de processos	129
6.1.- Classes do repositório de instâncias de processos	143
6.2.- Restrições aplicadas às tarefas.....	144
6.3.- Descrição das classes do serviço de agentes	147
6.4.- Descrição dos estados de um processo	148
6.5.- Descrição dos estados de uma tarefa	150
7.1.- Descrição das classes do repositório de artefatos	171
8.1.- Relação de tarefas, entradas e saídas do processo de solução de problemas.....	175
A.1 – Relacionamentos de entrada.....	199
A.2 – Relacionamentos de saída.....	200
A.3 – Classes pré-definidas da PML PROMENADE.....	210
A.4 – Estereótipos UML para a modelagem de processos.....	217
A.5 – Mapeamento elementos X associações.....	218
A.6 – Estereótipos para gráficos.....	225

LISTA DE SIGLAS E ABREVIATURAS

AMBGES	Ambiente Integrado para o Apoio ao Desenvolvimento e Gestão de Projetos de Software para Sistemas de Controle de Satélite
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integrated
E ³	Environment for Experimenting and Envolving Software Process
FIPA	Foundation for Intelligent Physical Agents
J2EE	Java 2 Enterprise Edition
JADE	Java Agent Development Framework
MOF	Meta Object Facility
OMG	Object Management Group
PDE	Process Definition Environment
PML	Process Modeling Language
PROMENADE	Process-oriented Modelling and Enactment of Software Developments
PSEE	Process-Centered Software Engineering Environment
SEI	Software Engineering Institute
SPI	Software Process Improvement

SPICE	Software Process Improvement and Capability Determination
UML	Unified Modeling Language
UPM	Unified Process Model
WFMC	Workflow Management Coalition

CAPÍTULO 1

INTRODUÇÃO

"A qualidade de um produto de software é determinada pela qualidade do processo utilizado para seu desenvolvimento e manutenção " (Kellner e Hansen,1988).

A citação ora apresentada procura alertar sobre a importância de formalizar e melhorar os processos de software organizacionais (Kellner e Hansen, 1988) (Humphrey e Kellner, 1989), com o intuito de ganhar qualidade e produtividade, fornecendo melhores produtos e serviços a seus clientes.

No desenvolvimento de software atual, pessoas trabalham de forma cooperativa, executando e coordenando suas tarefas. Isso pode ser feito de maneira informal quando elas são simples e pequenas, porém, para um grande projeto, assim como uma fábrica, surge a necessidade de um maior formalismo devido as exigências organizacionais (Humphrey e Kellner, 1989).

Dentro desse contexto, o conceito de processo (PMI, 2000) (SPICE, 1993) entra em cena para formalizar e especificar a condução do desenvolvimento de software, que pode ser realizado de forma caótica em muitas organizações.

Em muitas organizações, a qualidade de um produto de software depende exclusivamente de esforços individuais, da capacitação da equipe de desenvolvimento, do conhecimento de uma nova técnica aplicada, e das características do produto que ficam retidas com a pessoa responsável por sua construção e não com a organização. No intuito de minimizar esses efeitos adversos e obter um maior controle da qualidade do que se produz, algumas organizações investem tempo e dinheiro com esforços de melhoria de processos.

Os benefícios alcançados com esforços de melhoria de processos são inúmeros. Isso pode ser avaliado tanto pelos aspectos financeiros, referindo-se

a economia de recursos, tempo, custo, dentre outros fatores, quanto pelo que diz respeito à qualidade e produtividade de produtos.

Nos últimos anos, esforços vêm sendo gastos pela comunidade de engenharia de software no sentido de se desenvolver e elaborar modelos de qualidade, onde se aborda a melhoria de processos sob o aspecto gerencial. Esses modelos apresentam a importância de caracterizar formalmente os processos, dessa maneira, a realização de uma avaliação torna-se possível. Posteriormente, são levantados diversos aspectos referentes aos processos, esses aspectos são avaliados e uma lista de ações é proposta para a melhoria.

O (CMM) *Capability Maturity Model* (SEI, 1993), um modelo que avalia a maturidade dos processos de uma organização, procura atribuir um nível de maturidade à organização, onde o aumento dessa maturidade se dá por meio da avaliação das práticas-chave implementadas pela organização. Essas práticas, encontradas em áreas-chaves de processos, são organizadas em cinco níveis de maturidade. Para a evolução do nível, deve-se implementar todas as práticas encontradas no nível a ser evoluído mais as práticas do nível em que se encontra a organização.

Outros modelos como o *Software Process Improvement and Capability dEtermination* (SPICE), 1993, hoje norma ISO 15504 e o (CMMI) *Capability Maturity Model Integrated* (SEI, 2000) descrevem processos necessários para atingir a maturidade no desenvolvimento, categorizados em áreas de processo, onde cada processo implementado, pode ser avaliado por meio de seis níveis de maturidade.

As organizações que desenvolvem software, vêm adotando, em escala crescente, a implementação destes modelos, porém, devido a dificuldade de formalização e implementação de processos, pesquisas vem sendo realizadas em busca de soluções sob uma perspectiva tecnológica.

A área de modelagem de processos de software, (Kelner e Humphrey, 1988) (Humphrey e Kellner, 1989) (Ben Shaul et. al., 1994) (Jaccheri et. al., 1998) (Georgakopoulos e Hornick, 1995) (Fuggeta, 2000), encontra-se evoluindo vertiginosamente nos últimos anos no sentido de conseguir minimizar os esforços referentes a elicitação ou descoberta dos processos adotados e projeto de novos processos, o que vem auxiliando muito a formalização de um processo de software.

Em conjunto com a área de modelagem de processos, encontra-se em desenvolvimento a área de apoio à execução de processos, do termo em inglês *software process enactment* em Ambientes de Engenharia de Software Centrados em Processo (*PSEE - Process Centered Software Engineering Environments*) (Christie, 1993) (Ben Shaul et. al. 1994) (Ellmer, 1995) (Ambriola et al., 1997) (Fuggeta, 2000).

Os Ambientes de Engenharia de Software, apresentam-se como uma solução tecnológica para o efetivo apoio a esforços de melhoria de processos. Os PSEE's, como uma nova classe de ambientes, consideram a Engenharia de Software como a execução de um processo definido explicitamente.

Sant'Anna (2000) redefine ambiente de engenharia de software como um conjunto de recursos flexíveis e abrangentes, dando suporte eficiente e de forma integrada ao processo de engenharia de software definido para uma organização.

A tecnologia de processos de software propõe o desenvolvimento e adoção de PSEEs para automatizar a gerência dos processos (Gimenes, 1994). Esta tecnologia traz benefícios, como por exemplo: melhor comunicação entre as pessoas envolvidas e consistência do que está sendo feito; realização de algumas ações automáticas, permitindo que os seus usuários não realizem tarefas repetitivas; fornecimento de informações sobre o andamento do processo quando necessário; possibilidade de reutilização de processo de

software e a coleta automática de métricas importantes para o controle e aperfeiçoamento de processos (Reis, 2001).

Em meio a este cenário surge o PSEE e-WebProject® (Sant'Anna et al., 2002), um produto comercial originado do trabalho de pesquisa "Um Ambiente Integrado para o Apoio ao Desenvolvimento e Gestão de Projetos de Software para Sistemas de Controle de Satélite - AMBGES" (Sant'Anna, 2000) desenvolvido no Instituto Nacional de Pesquisas Espaciais - INPE. O principal objetivo do ambiente é fornecer um conjunto de ferramentas integradas para trabalho cooperativo, suporte à execução de processos de engenharia de software, gestão de projetos, gestão do conhecimento organizacional e apoio à infra-estrutura organizacional.

O e-WebProject, ainda em desenvolvimento, utiliza tecnologia Java (J2EE) e é disponibilizado através da WEB/Internet, o que permite uma maior e mais fácil disseminação das informações. Este trabalho de pesquisa tem foco na definição e desenvolvimento de um serviço de coordenação de processos que será integrado ao PSEE e-WebProject.

A principal função deste serviço é o apoio à gestão de processos de software, que devem ser apoiados pelo ambiente. Entende-se por processos de software:

"Um conjunto de atividades inter-relacionadas, que transformam entradas em saídas". (ISO, 1995).

1.1.- Motivação

O que motiva a maioria das pessoas, principalmente a gerência de uma organização a adotarem práticas de esforços de melhorias de processos, *SPI Software Process Improvement*, pode ser avaliado em termos de benefícios. Apresenta-se nesta seção alguns ganhos obtidos com a adoção de práticas de SPI.

Observa-se, já em análises preliminares, os benefícios alcançados com esforços de melhorias de processos. Um relatório técnico publicado pelo *Software Engineering Institute SEI*, (Herbsleb et. al. 1994) relata alguns resultados, ainda que iniciais, dos ganhos obtidos de organizações que adotaram práticas de SPI baseadas no modelo CMM. Nele encontram-se coletados, analisados e apresentados: os dados de 13 organizações que adotaram o modelo; os resultados sobre os custos e ganhos obtidos com esforços de SPI, dentre os quais podem-se citar: a melhoria anual de produtividade, detecção de defeitos, tempo de mercado e relatórios de defeitos pós entrega de produtos. Um resumo dos dados analisados pelo SEI pode ser visto na Tabela 1.1, onde as organizações que forneceram os dados se encontravam nos mais variados níveis de maturidade, classificadas dentro de faixas de investimentos e de médias de benefícios obtidos com a adoção dos esforços de SPI.

TABELA 1.1.- Custos e ganhos de 13 organizações que adotaram práticas de esforços de melhorias de processos baseados no modelo CMM.

CATEGORIA	FAIXA	MÉDIA
Total anual de custos com atividades de SPI	US\$ 49.000 – US\$ 1.202.000	US\$ 245.000
Anos gastos com esforços de SPI	1 - 9	3,5
Ganhos em produtividade obtidos por ano	9% - 67%	35%
Redução anual em tempo de mercado	15% - 23%	19%
Redução anual em relatórios de defeitos pós entrega	10% - 94%	39%
Valor retornado por cada dólar investido em esforços de SPI	4.0 - 8.8	5.0

FONTE: Herbsleb et. al. (1994, p. 15).

Em um estudo mais recente (El Eman e Briand, 1999), são apresentados resultados referentes a custos e benefícios de organizações que adotaram práticas de melhoria de processos usando diferentes modelos, dentre eles estão o próprio CMM, o modelo SPICE, ISO 9001 e outras.

A tabela 1.2 apresenta um resumo de resultados de investigações realizadas sobre custos com atividades de SPI usando diferentes abordagens analíticas e

de *benchmarking*. A tabela mostra a organização e respectivo programa de SPI, os custos e benefícios alcançados com a adoção do programa.

TABELA 1.2.- Experiências organizacionais ilustrando os custos e benefícios da adoção de programas de SPI.

Organização e Programa de SPI	Custos	Benefícios
<ul style="list-style-type: none"> • Esforços de SPI na divisão de Engenharia de Software da Hughes Aircraft • A divisão possuía 500 empregados na época 	<ul style="list-style-type: none"> • O custo da própria avaliação estava em US\$ 45.000 • 2 anos de implantação do programa de SPI custaram em torno de US\$ 400.000 • Implementação de um plano de ação para mover do 1o nível de maturidade para o 2o levou em torno de 18 meses 	<ul style="list-style-type: none"> • Algo em torno de US\$ 2.000.000 de economia de custos • Os benefícios foram calculados em torno de 5 vezes os valores gastos com a implantação do programa • Houve melhora do nível de vida no trabalho (Os engenheiros de software precisaram trabalhar fora de horário poucas vezes)
<ul style="list-style-type: none"> • Esforços de SPI implantados no Schlumberger Laboratory for Computer Science 	<ul style="list-style-type: none"> • Grande centros de engenharia (em torno de 120 a 180 engenheiros), utilizaram de 1 a 5 pessoas em tempo integral em atividades de SPI. Centros menores (em torno de 50 a 120 engenheiros) alocaram em média 3 pessoas em tempo integral em atividades de SPI 	<ul style="list-style-type: none"> • Comunicação em projeto melhorada. • Relatos de clientes que confirmaram a melhora na qualidade do produto. • Um grupo melhorou o tempo de mercado, reduzindo ciclos de validação de requisitos de 15 para 34. • Um grupo mais que dobrou a sua produtividade. • Um grupo aumentou a porcentagem de projetos completados no prazo de 51% para 94% • Um grupo reduziu pela metade a densidade de defeitos de seus produtos
<ul style="list-style-type: none"> • Dados coletados de 33 empresas, utilizando questionários e/ou entrevistas 	<ul style="list-style-type: none"> • Os autores apresentaram exemplos de dados referentes a custos com atividades de SPI. • Por exemplo, algumas organizações aumentaram de 7% para 8% o total de esforço em coleta de dados, e aumentaram em 2% os custos com o projeto, focando no projeto, reduzindo defeitos. 	<ul style="list-style-type: none"> • Algumas organizações aumentaram a produtividade, reduziram os níveis de defeitos, reduziram os esforços com retrabalho, reduziram custos e melhoraram as estimativas para completar os projetos. • Outros benefícios incluem, menor retorno de empregados e aumento da cooperação entre grupos funcionais.
<ul style="list-style-type: none"> • Esforços de SPI em nível corporativo iniciados em 1992 na AlliedSignal 	<ul style="list-style-type: none"> • Usando dados do SEPG (Software Engineering Process Group), foram medidos investimentos 	<ul style="list-style-type: none"> • Um local obteve um aumento de produtividade de 7 para 1 no tempo de calendário para gerar documentos em torno de 1000 páginas com uma

(continua)

Tabela 1.2 – Conclusão.

Aerospace	em 8 locais.	redução de 50% no custo por página. <ul style="list-style-type: none"> • Deficiência de relatórios na documentação foi reduzida para zero. • Um local relatou que a manutenção de Linhas de Código (LOC) por pessoa foi reduzido em 50% e o tempo de teste foi reduzido para 60% sem um aumento evidente de defeitos entregues.
<ul style="list-style-type: none"> • Organização é o Software Systems Laboratory em Raytheon, 400 engenheiros de software empregados. • Iniciativas de SPI iniciados em 1988: resultados relatados após 5 anos • A organização progrediu do nível 1 para o nível 3 neste período. 	<ul style="list-style-type: none"> • US\$ 1 milhão investido por ano 	<ul style="list-style-type: none"> • Um retorno de 7.7 para 1, em cada dólar investido. • Custos com retrabalho de todos os projetos foram reduzidos de 41% para 11%. • Mais projetos finalizados no prazo e custos estabelecidos no orçamento. • Aumento de produtividade em um fator de 2.3 em 4.5 anos • Engenheiros de Software gastaram menos noites e fins de semana no trabalho e tiveram uma melhora no nível de vida.

FONTE: (El Eman e Briand, 1999, p. 3).

Tornam-se evidentes os ganhos obtidos com as práticas de SPI. A adoção dessas práticas pode ser potencializada com o uso de Ambientes de Engenharia de Software Centrados em Processos que fornecem um apoio tecnológico, sistematizando as atividades relacionadas a atividades de SPI, melhorando a cooperação entre os vários envolvidos nos projetos, fornecendo subsídios para a gerência para a tomada de decisões, dentre outros fatores. Esse é o principal fator de motivação para dar andamento a este trabalho.

1.2.- Objetivo do Trabalho de Pesquisa

Atualmente o e-WebProject possibilita a implantação de processos automatizados, porém, esta implantação envolve o desenvolvimento do aplicativo específico que deseja automatizar, isto ocorre, devido à falta de mecanismos do ambiente para o apoio a gestão de processos. Essa abordagem gera alguns transtornos por causa do dinamismo dos processos de software, a necessidade de mudança do processo acarreta em novo desenvolvimento.

Este trabalho de pesquisa tem como objetivo principal, a especificação, modelagem, projeto e construção de um serviço que apóie a gestão de processos e que seja integrado ao PSEE e-WebProject. No contexto da gestão de processos encontram-se: a definição de processos de software e o apoio à execução destes processos. Portanto, menciona-se para o escopo deste serviço facilidades para:

- 1) o apoio à definição de processos;
- 2) o apoio à execução dos processos;
- 3) o acompanhamento da execução dos processos;
- 4) o controle de artefatos produzidos em processos.

1.3.- Estrutura do Trabalho

Esta dissertação encontra-se organizada em mais oito capítulos e cinco apêndices, conforme descrição a seguir:

- **CAPÍTULO 2 – PROCESSOS DE SOFTWARE E MODELAGEM DE PROCESSOS** - aborda o tema processos de software e modelagem de processos: suas definições, importância, razões para modelar processos, elementos de processos e linguagens de modelagem de processos.
- **CAPÍTULO 3 – AMBIENTES DE ENGENHARIA DE SOFTWARE CENTRADOS EM PROCESSO E AUTOMAÇÃO DE PROCESSOS** – apresenta: as características de ambientes de engenharia de software centrados em processo; as características do PSEE e-WebProject, ambiente onde se insere este trabalho; e as tecnologias que servem de base para automação de processos de software utilizadas neste trabalho: a tecnologia de workflow e a tecnologia de agentes.

- **CAPÍTULO 4 – O SERVIÇO DE COORDENAÇÃO DE PROCESSOS** - descreve uma visão geral do serviço do ambiente e-WebProject responsável pelo apoio aos processos de software definidos no ambiente; apresenta uma visão sobre o escopo do serviço, seus requisitos, componentes, sub-módulos e arquitetura de desenvolvimento.
- **CAPÍTULO 5 – O AMBIENTE DE DEFINIÇÃO DE PROCESSOS DE SOFTWARE** - apresenta a modelagem do Ambiente de Definição de Processos, sub-módulo do serviço de coordenação de processos, responsável pelo apoio à definição de processos de software no ambiente e-WebProject.
- **CAPÍTULO 6 – A MÁQUINA DE PROCESSOS** – apresenta a modelagem da máquina de processos, sub-módulo do serviço de coordenação de processos responsável pelo controle da execução de processos. A modelagem e o desenvolvimento desse sub-módulo são baseados em agentes autônomos colaborativos. Apresentam-se aspectos de modelagem estruturais e comportamentais da máquina de processos.
- **CAPÍTULO 7 – O REPOSITÓRIO DE ARTEFATOS** - apresenta a modelagem do repositório de artefatos, sub-módulo do serviço de coordenação de processos responsável pelo controle dos artefatos produzidos durante a execução dos processos.
- **CAPÍTULO 8 – UM PROTÓTIPO DO SERVIÇO:** apresenta um protótipo inicial desenvolvido para o serviço de coordenação de processos, onde se utiliza a modelagem de um processo de software, como um Estudo de Caso para a utilização do serviço.

- **CAPÍTULO 9 – CONSIDERAÇÕES FINAIS:** Neste capítulo são apresentadas considerações sobre o trabalho, conclusões, contribuições e perspectivas sobre trabalhos futuros.
- **APÊNDICE A – ALGUMAS ABORDAGENS PARA A MODELAGEM DE PROCESSOS** – neste apêndice são descritos cinco abordagens para a modelagem de processos: a descrição das PMLs: ProNet, E3 e PROMENADE; o UPM (Unified Process Model), uma submissão inicial da OMG para a padronização da modelagem de processos de software; e uma abordagem para a utilização da UML como linguagem para a modelagem de processos.
- **APÊNDICE B – O CÓDIGO FONTE DO AGENTE MENSAGEIRO** - neste apêndice é apresentado o código fonte de agente mensageiro baseado na plataforma JADE.
- **APÊNDICE C – DESCRIÇÃO DOS CASOS DE DIAGRAMAS DE SEQUENCIA DO AMBIENTE DE DEFINIÇÃO DE PROCESSOS** – Neste apêndice são detalhados o conjunto de casos de uso e os diagramas de seqüência do Ambiente de Definição de Processos apresentada no Capítulo 5.
- **APÊNDICE D – DESCRIÇÃO DOS CASOS DE USO DA MÁQUINA DE PROCESSOS** - Neste apêndice é descrito em detalhes o conjunto de casos de uso da Máquina de Processos apresentada no Capítulo 6.
- **APÊNDICE E – DESCRIÇÃO DOS CASOS DE USO DO REPOSITÓRIO DE ARTEFATOS** - Neste apêndice são detalhados o conjunto de casos de uso do repositório de artefatos apresentado no Capítulo 7.

CAPÍTULO 2

PROCESSOS DE SOFTWARE E MODELAGEM DE PROCESSOS

A pesquisa sobre processos de software tem ganhado força nos últimos anos devido a uma crescente demanda por desenvolvimento software em todo o globo, para os mais variados ramos de atividade, em destaque, pode-se citar: a Indústria de Tecnologia da Informação, a Área Militar e o Setor de Telecomunicações.

Nesse contexto, muitas organizações que desenvolvem software, necessitam melhorar a qualidade dos produtos que desenvolvem, devido às exigências do mercado e ao crescente aumento do nível de criticidade e complexidade atribuídos aos produtos de software atuais.

Isso significa que o software deve ser desenvolvido de uma maneira previsível, isto é, todas as etapas e produtos que um processo gera devem ser conhecidos de uma forma explícita. Em muitas organizações o processo de desenvolvimento encontra-se na cabeça das pessoas envolvidas.

Um outro ponto de vista (Osterweil, 1987), parte da observação que todas as organizações são diferentes. Diferem na cultura, nas habilidades das pessoas, nos produtos entregues, na área comercial e nas estratégias de desenvolvimento. Além de dentro da mesma organização, existirem uma certa quantidade de projetos diferentes com características particulares.

As organizações que desenvolvem software procuram desenvolver produtos diferentes com características específicas. Conseqüentemente, não há nenhum processo original, “pronto para uso”, de desenvolvimento de software. O processo deve ser definido baseado no problema a ser resolvido; deve ser

adaptado ao projeto, respeitar as exigências dos clientes, as particularidades da organização e do produto desenvolvido.

Também, os ambientes que suportam o desenvolvimento de software devem ser adaptados ao processo específico definido para aquela organização. Para satisfazer a estes objetivos, Osterweil (1987) concluiu, nós precisamos de linguagens para descrever processos de desenvolvimento.

A produção e manutenção de software requer organização própria; por exemplo, estruturas e condições para apoiar as interações entre as pessoas envolvidas na produção de um software e entre pessoas e ferramentas computadorizadas (Conradi et. al., 1992). Esse processo organizacional é denominado no contexto do desenvolvimento de software como processo de software.

Em (Sant'Anna, 2000) pode-se verificar que nos primeiros anos da Engenharia de Software o termo processo era utilizado para denotar o processo de desenvolvimento de software como um todo, ou seja, um processo macro e abstrato que levava à construção do software. Com o surgimento dos modelos de qualidade (SEI, 1993) (SPICE, 1003) (SEI, 2000) (ISO, 1995) outros termos relacionados a processo, como: processo de gerenciamento, processo de garantia da qualidade, etc., foram apresentados também como responsáveis e importantes na construção do produto de software final.

Um processo de software inclui atividades técnicas e administrativas que são utilizadas na produção e manutenção de software, por exemplo: determinação e especificação de sistema e requisitos de software, análise e administração de riscos, prototipação de software, *design*, implementação, verificação e validação, controle de qualidade de software e garantia, integração de componentes, documentação, gerenciamento de versões e configuração de software, gerenciamento de dados, evolução do software, administração de projeto, etc. (Madhavji, 1991).

A exploração do conceito de processo de software torna-se importante na medida em que se procura entendê-los, desta forma, consegue-se subsídios ao desenvolvimento de um mecanismo coordenador de processos, elemento essencial ao apoio à gestão de processos no ambiente e-WebProject.

2.1.- Definições

Diversas são as definições para processo de software, entre elas destacam-se:

“Uma série de ações para se obter um resultado” (PMI, 2000).

“Um processo de software composto de um conjunto de processos capazes de conduzir a organização envolvida com a construção de produtos de software com qualidade e custos previsíveis, de forma eficiente, gerenciada e com a possibilidade de melhoria constante” (Sant’Anna, 2000).

"Processo de Software é um conjunto de atividades parcialmente ordenadas para gerenciar, desenvolver e manter um sistema de software" (Acuña, 2001).

“O processo de software é uma seqüência de estágios para desenvolver ou manter o software, apresentando estruturas técnicas e de gerenciamento para o uso de métodos e ferramentas, e incluindo pessoas para as tarefas do sistema” (Humphrey, 1995).

“Processo é um conjunto de atividades organizadas que transformam entradas em saídas” (Kontonya e Sommerville, 2000).

“É o conjunto de atividades, métodos, práticas e transformações que as pessoas empregam para desenvolver e manter software e os produtos associados (por exemplo, planos de projeto, documentos de projeto/design, código, casos de teste, manual do usuário) (Ministério da Ciência e Tecnologia do Brasil, 1999)”.

“Um grupo parcialmente ordenado de subprocessos, com grupos de artefatos relacionados, recursos humano e computadorizados, estruturas

organizacionais, pretendidos a produzir e manter o desenvolvimento de um dado software requisitado” (Lonchamp, 1993).

“O processo ou um conjunto de processos, utilizados por uma organização ou projeto para: planejar, gerenciar, executar, monitorar, controlar e melhorar as atividades relacionadas à construção de software”. (SPICE 1993)

Neste trabalho utiliza-se, a definição apresentada na norma ISO/IEC 12207 (ISO, 1995) para o termo “processo de software”:

“Um conjunto de atividades inter-relacionadas, que transformam entradas em saídas”

O termo “processo de software” também é denominado, quando o mesmo não caracteriza confusão terminológica no meio utilizado, apenas como "processo".

Um processo, por sua vez, é composto por outros elementos. Não existe um consenso sobre a taxinomia dos elementos que compõem um processo de software. Os trabalhos sobre o assunto divergem entre si, diversos esforços (Conradi et. al.,1992) (Feiler et. al., 1991) (Dowson et. al., 1991) (Madhavji, 1991) (Lonchamp, 1993) foram e estão sendo realizados para tentar encontrar um consenso entre as terminologias adotadas. O que pode ser observado (Lonchamp, 1993) é que qualquer processo de software, de forma geral, possui como principais componentes: subprocessos, recursos humanos e computacionais, metas e estruturas organizacionais.

2.2.- Porque Processos de Software São Importantes

Na década passada, houve um aumento no interesse de melhorar a qualidade na maioria dos setores industriais. Além disso, houve uma consciência crescente da importância central dos processos de produção. Os processos são importantes porque a indústria se importa com suas qualidades intrínsecas, tais como a uniformidade dos comportamentos através dos diferentes projetos e a produtividade, melhorar o tempo de chegada ao mercado *time-to-market* e

redução dos custos de produção. Mas, eles são também importantes porque a experiência mostrou que os processos têm uma influência profunda na qualidade dos produtos; isto é, controlando processos pode-se conseguir um controle melhor da qualidade solicitada aos produtos.

Isto é aplicável também nas fábricas de software devido à natureza intrínseca do software. Se um processo explícito estiver no lugar, o desenvolvimento do software prossegue de uma forma sistemática e em ordem. Isto impede que os erros estejam introduzidos no produto e fornece meios para o controle da qualidade do que está sendo desenvolvido. A Figura 2.1 fornece uma visão intuitiva para reforçar este ponto.

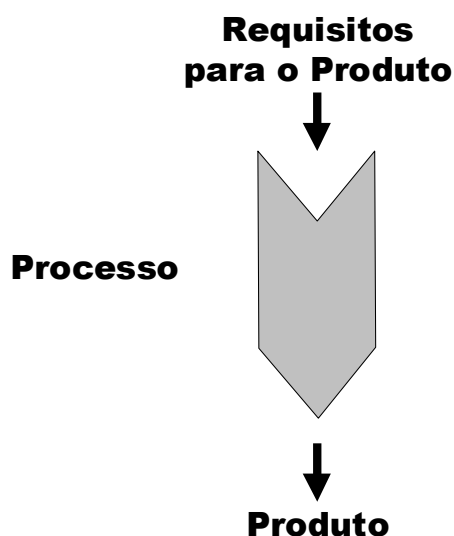


FIGURA 2.1.- O processo como uma caixa preta (1).

FONTE: Cugola e Ghezzi (1998).

Se não existir nenhuma noção explícita dos processos adotados, o desenvolvimento do produto pode ser considerado como uma caixa preta, onde os únicos fluxos visíveis estão nas extremidades: entrada e saída. No lado da

entrada, o fluxo para a entrada do processo representa as exigências (necessidades) do produto. Na saída, a outra extremidade do processo, espera-se de forma esperançosa, que o produto desejado seja entregue.

Infelizmente, em muitos casos práticos, o produto desejado não aparece nos meses ou anos que se espera a entrega (saída), geralmente, com investimento de muito dinheiro.

Quando o produto é finalmente entregue (saída do processo), o desenvolvimento é freqüentemente atrasado e também torna-se caro importar-se com qualidade. Em consequência disso, torna-se necessário que o interesse para a qualidade permeie todo o processo, isto é, não pode haver atraso ao fim do desenvolvimento.

Torna-se clara a ineficiência dos processos vistos como na abordagem apresentada pela Figura 2.1. A primeira e principal dificuldade encontrada está em se fazer a eliciação dos requisitos, como mostrado na visão do processo de desenvolvimento de software apresentado na Figura 2.2.

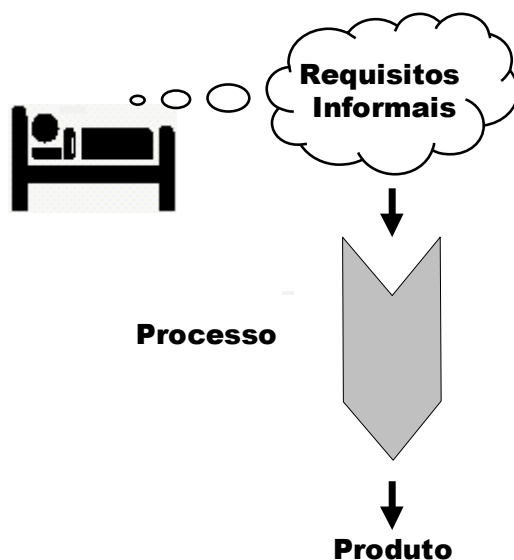


FIGURA 2.2.- O processo como uma caixa preta (2).

FONTE: Cugola e Ghezzi (1998).

De acordo com tal visão, o desenvolvimento de um produto começa (na maioria dos casos) com alguns requisitos extraídos de maneira informal, que se originam no mundo do negócio dos clientes. Esta informalidade, pode acarretar alguns transtornos, já que não se tem uma maneira sistemática de descobrir e traduzir essa informação em requisitos precisos. Isso ocorre, porque em muitos casos, o cliente não sabe exatamente o que quer. O cliente tem uma percepção de seus problemas, mas não é capaz de traduzir esta percepção em requisitos precisos. Consequentemente, os requisitos de entrada para o processo são provavelmente muito informais, nebulosos, e em sua maior parte incompletos, talvez contraditórios, não refletindo assim as necessidades reais dos usuários. Se o processo de desenvolvimento for estruturado como uma caixa preta, não há nenhuma visibilidade sobre o que está fluindo e tão menos sobre como progride o processo de desenvolvimento. Eventualmente, quando o produto é entregue, é muito provável que o produto não atenda as expectativas do cliente. Embora o software seja mais fácil de modificar do que outros tipos de artefatos tradicionais, como uma cadeira por exemplo, o custo das modificações de desenvolvimento é muito elevado e sua eficácia é freqüentemente muito baixa.

Deste modo, pode-se dizer que é extremamente arriscado basear todas as decisões do projeto na suposição de que os requisitos iniciais adquiridos pelo Engenheiro de Software, capturaram fielmente as expectativas dos clientes.

Para reduzir os riscos, é necessário abrir a caixa preta. Deve-se definir um processo apropriado que forneça a visibilidade do que está sendo desenvolvido. Dentro da caixa preta, pode-se esperar validar o que está sendo desenvolvido de encontro com as expectativas dos clientes.

O processo pode assim, fornecer um “feedback” contínuo aos desenvolvedores. Isto pode reduzir o tempo para a tomada de uma decisão e o tempo para descobrir que uma decisão tomada estava errada, reduzindo

assim, os custos necessários para desenvolver um produto de software aceitável.

Uma outra dificuldade encontra-se na inevitável tendência de mudança dos requisitos durante o processo de desenvolvimento. Como mencionado, os clientes não sabem exatamente o que querem, por causa disso, sustenta-se que os requisitos mudam durante a execução do processo.

Um exemplo comum que resulta em mudança rápida de requisitos de desenvolvimento, acontece com softwares para Internet, mais precisamente aplicações de e-commerce, nesses casos, os requisitos iniciais são conhecidos somente parcialmente e não há nenhum cliente para fornecer uma lista precisa das características que a aplicação deve fornecer.

Novas demandas aparecem quando clientes potenciais utilizam os protótipos entregues e, após um dado período de utilização, estes usuários iniciais da aplicação fornecem um *feedback* aos engenheiros de software e desenvolvedores que posteriormente vão melhorar o produto. Estes conceitos sugerem um esquema de processo alternativo, que seja descrito detalhadamente.

Como é sugerido pela Figura 2.3, um processo de produção transparente permite que o cliente compreenda o que está sendo produzido pelo processo, isto significa que o cliente observe os artefatos produzidos durante a execução do processo. Tais artefatos (por exemplo, casos do uso, protótipos intermediários, versões preliminares e incompletas, documentação do projeto, definição do caso do teste, etc..) podem ser usados para validação da etapa do processo atual. Em consequência da validação, é possível decidir-se deve-se prosseguir à etapa seguinte, ou re-iterar a etapa precedente.

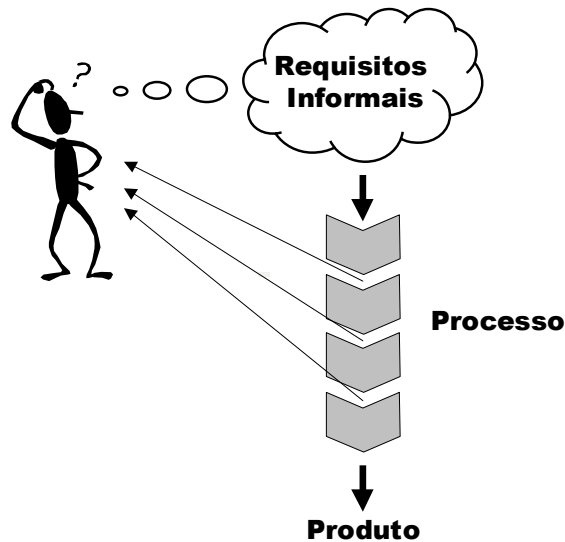


FIGURA 2.3.- Um processo transparente.

FONTE: Cugola e Ghezzi (1998).

Outra razão fundamental para dizer que um processo de caixa-preta como o apresentado na Figura 2.1, se torna inaceitável para projetos de desenvolvimento de software, está no fato de que não se pode avaliar a qualidade de um produto, simplesmente olhando para ele mesmo. Isto pode ser diferente para outros tipos de artefatos como uma ponte, por exemplo.

No caso da ponte, a qualidade pode ser medida por meio de algumas condições físicas, como a carga de um determinado peso à ponte, se ela sustentar a carga ao qual ela foi projetada, a qualidade está certificada.

No caso de sistemas de software, esta regra não se aplica, por exemplo, se um determinado produto de software passar com sucesso pela execução de um determinado teste, isto não garante, que o mesmo produto tenha o mesmo comportamento em diferentes situações, ou que não haja defeitos no produto entregue.

Para se conseguir uma maior confiabilidade na corretividade do produto de software entregue, recomenda-se, que os processos utilizados durante o desenvolvimento do produto de software, sejam estruturados de um modo que

faça com que o desenvolvimento seja conduzido de forma sistemática de modo a reduzir os defeitos. Combinando prevenção, verificação contínua e validação, um processo explícito e claramente definido pode melhorar a qualidade do software e a produtividade dos engenheiros e desenvolvedores.

2.3.- Modelagem de Processos de Software

2.3.1.- Definição de Modelo de Processo

Um processo é o principal elemento de um ambiente de desenvolvimento de software já que a qualidade de um produto de software está diretamente ligada a qualidade do processo utilizado para seu desenvolvimento (Kellner, 1988) (Kelner e Humphrey, 1989). Surge então a necessidade de formalizar os processos organizacionais pelos quais são desenvolvidos os produtos de software. A modelagem de processos constitui-se em um recurso importante para alcançar tal formalização. Neste caso, um modelo de processo é uma representação abstrata de um processo de software (Acuña, 2001).

2.3.2.Razões para se Modelar um Processo

Os processos precisam ser formalizados ou padronizados de forma a se entender o seu funcionamento, isso possibilita o melhor treinamento, as propostas para melhorias dentre outros fatores. Um modelo de processo deve especificar os pré-requisitos e conseqüências de cada tarefa, bem como a sincronização entre essas tarefas (Ben-Shaul, 1994).

A modelagem de um processo de software deve ser conduzida de modo a possibilitar o entendimento e a padronização do processo, para tanto deve-se observar os principais objetivos para se modelar o processo de software (Kellner et. al., 1988) (Humphrey, 1989) (Fuggeta, 2000) que são:

- **Entendimento do processo** - Um modelo de processo pode ser utilizado para representar, de uma forma precisa como o processo está

estruturado e organizado, tornando-se um instrumento importante para a eliminação de inconsistências na especificação do processo;

- **Projeto de um processo** - Um modelo de processo pode ser utilizado para o projeto de um novo Processo, descrevendo sua estrutura e organização;
- **Treinamento e Educação** - Uma descrição de um processo, pode ser útil para o treinamento e aprendizado de equipes sobre procedimentos organizacionais;
- **Simulação e otimização de processos** - Uma descrição de um processo pode ser útil para a avaliação de problemas e propostas para melhorias;
- **Suporte a processos** - Uma descrição detalhada de um processo pode ser utilizada para fornecer diferentes níveis de suporte, para pessoas envolvidas no processo.

Segundo Humphrey e Kellner (1989), a utilização de processos padronizados, faz com que cada experiência de projeto contribua para a melhoria dos processos na organização, fornecendo uma base estrutural para medição. No entanto, definições de processo levam tempo e esforço, o que torna impraticável novas definições de processo para cada projeto.

A modelagem de processos de software, iniciou-se através da utilização de técnicas de análise de sistemas, como a utilização de diagramas do paradigma estruturado da Engenharia de Software (utilização de DFDs por exemplo) para se representar um processo (Kellner e Hansen, 1988), (Humphrey e Kellner, 1989), outros autores utilizam atualmente modelos Orientados a Objetos (Jacobson et. al., 1999a; 1999b) (Georgakopoulos e Hornick, 1995), por fim surgiram as chamadas (PML) linguagens de modelagem de processos, que

agregam os diversos elementos utilizados nas formas de modelagem anteriores.

As PMLs são constituídas de elementos próprios (*core*) com semânticas próprias que representam os elementos comuns de processos e um conjunto de associações que são utilizadas para a construção do modelo (Jacobson et. al., 1999a) (Humphrey e Kellner., 1989) (ISO,1995).

Geralmente o processo é caracterizado por meio de um diagrama com notação gráfica pertencente à linguagem, onde são estruturados os elementos e a forma como eles serão executados "*process enactment*".

A modelagem das informações em um modelo de processo pode ser estruturada de diferentes pontos de vista. Uma lista de *perspectivas de informação* para a modelagem do modelo de processo (Acuña, 2001) é apresentada a seguir:

- **Funcional** - representa quais elementos de processo estão sendo implementados e quais fluxos de informação são importantes;
- **Comportamental** - representa quando (seqüência) e sob quais condições os elementos de processo são implementados;
- **Organizacional** - representa onde e por quem os elementos de processo são implementados;
- **Informativo** - representa as entidades de informação que são manipuladas por um processo, incluindo sua estrutura e relacionamentos.

Os modelos são construídos de modo a representar informações específicas, levando-se em conta estas características. A maioria das abordagens mencionadas, não converge para todas as classes de informações, sendo necessário à modelagem sob diferentes aspectos.

Um modelo de processo deve ser executado em um ambiente de engenharia de software. Um modelo de processo executável deve representar sua estrutura e comportamento (Shleicher et. al., 1998, 2000, 2001).

2.3.3.- Histórico

A área de modelagem de processos de software, do inglês "software process modelling", (Kellner e Hansen, 1988) (Humphrey e Kellner, 1989) (Ben Shaul et. al. 1994) (Jaccheri et. al. 1998, 1999) (Georgakopoulos e Hornick, 1995) (Fuggeta, 2000), está evoluindo vertiginosamente nos últimos anos, no sentido de conseguir minimizar os esforços referentes a elicitação e representação dos processos adotados e projeto de novos processos.

Torna-se clara a necessidade da modelagem de processos de software, o ato de se modelar um processo, conhecido pela comunidade de engenharia de software como "software process elicitation", é um assunto de grande interesse, largamente pesquisado durante os últimos anos.

Vale salientar que, diversas técnicas não são recentes e são largamente utilizadas em metodologias para engenharia de requisitos, por exemplo. Processos de software possuem suas próprias peculiaridades e devido as suas características, as formas abordadas para modelagem até então, provocavam uma imensa confusão.

Kellner e Hansen (1988), abordam a modelagem de processos de software por meio da utilização de técnicas de análise de sistemas, onde eram utilizados diagramas de fluxo de dados (DFD), que forneciam apenas uma visão de como as tarefas eram executadas no processo.

Este tipo de modelo oferece uma visão limitada, mostrando apenas uma visão funcional do mesmo, descrevendo "o quê" estava sendo feito e qual dado estava fluindo, não oferecendo informações sobre precedência e consequência

sobre cada tarefa executada no processo, era necessário então, modelar o processo sob outros pontos de vista.

Outras linguagens como a STATEMATE (Kellner e Hansen, 1988), fornecia outros tipos de recursos, fornecendo subsídios para se modelar o processo sob outros dois pontos de vista: o comportamental, representado o comportamento (“quando” e “como”) através de diagramas de transição de estados e o estrutural, mostrando quem executa cada tarefa, representada por diagramas de módulo.

Outros autores pregam a utilização de modelos Orientados a Objetos para a representação de processos (Jaccheri et. al., 1998, 1999) (Georgakopoulos e Hornick, 1995) (Schleicher et. al., 1998, 2001) (Franch e Ribó, 1999a) (Mühlen e Becker, 1999). A linguagem unificada de modelagem (UML), por exemplo, oferece diversos diagramas que podem ser utilizados para modelagem de partes estáticas e dinâmicas de um processo. Alguns estudos convergem à extensão do meta-modelo UML para representar melhor os processos (Schleicher et. al. 1998, 2000, 2001) (Franch e Ribó, 1999a, 1999b).

As linguagens de modelagem de processos (PML) (Ben Shaul et. al., 1994) (Fuggeta, 2000) (Jaccheri et. al. 1998, 1999) surgiram posteriormente e conseqüentemente agregam diversos elementos característicos ao domínio dos processos de software. Dentre as principais, destacam-se a linguagem gráfica ProNet (Christie, 1993), representante da primeira geração de PMLs; e atualmente surgem representantes de uma nova geração de PMLs, destacando-se as linguagens E3 "*Environment for Experimenting and Envolving software processes*" (Jaccheri et. al. 1998, 1999) e PROMENADE "*Process-oriented Modelling and Enactment of Software Developments*" (Franch e Ribó, 1999b).

2.4.- Linguagens de Modelagem de Processo (Process Modeling Languages – PMLs)

Nesta seção, são abordados alguns aspectos referentes às linguagens de modelagem de processo, PMLs. A modelagem de Processo é uma área muito diversa e complexa. As necessidades para modelagem e execução apóiam tanto técnicas (por exemplo, expressividade, abstração, e multi-perspectivas) como não técnicas (por exemplo suporte comercial).

Existem muitas pesquisas e tecnologias que podem ser potencialmente adaptadas para o domínio da modelagem de processo. Portanto pode-se olhar para áreas relacionadas como: modelagem de informações, base de dados e grupos de trabalho *groupware*, para idéias e soluções operativas.

Uma PML expressa os processos de produção de software na forma de um modelo de processo, sendo uma descrição interna de computador de um processo externo. Essa descrição interna de computador, denota um modelo declarativo que permite uma possível interpretação por meio de um compilador, interpretador ou motor computacional, permitindo a execução do processo.

Nos últimos dez anos, muitas PMLs foram propostas, implementadas e testadas. Contudo, existe um pequeno acordo ou padronização, até mesmo nos conceitos e terminologias em uma tentativa de formalizar os conceitos básicos do domínio do processo de software e o conceito de meta-processo.

Estas duas dimensões configuram os requisitos básicos para uma PML: cada fase do meta-processo, por exemplo “elicitacão” (ato de extrair ou fazer surgir) deve ser apoiado por uma linguagem que permita expressar o modelo no nível de abstração apropriado; cada elemento de processo, por exemplo, tarefas, papéis, e produtos, precisam ser descritos.

Existe um acordo geral na identificação dos elementos primários de processo como atividades (tarefas), produtos, papéis, ferramentas, agentes, e suporte à

evolução. Outros elementos, desempenhando um papel secundário no contexto de um modelo de processo são espaços de trabalho (“workspaces”), visões de ferramentas, e visões de usuário. Finalmente, os modelos de cooperação, versão e transação, e modelos de qualidade têm de ser descritos.

Uma PML pode ser formal, semi-formal, ou informal. Uma PML formal possui uma sintaxe e semântica formal. As linguagens semi-informais normalmente têm uma notação gráfica com sintaxe formal, mas não uma semântica formal não sendo executável, por exemplo. As linguagens naturais, como o inglês ou português, podem ser usadas nas PMLs informais. Neste capítulo, dá-se ênfase sobre PMLs formais. Uma linguagem de especificação formal é uma linguagem cuja sintaxe, e semântica são formalmente definidas. Uma especificação formal é um objeto matemático que pode ser analisado usando métodos matemáticos. As PML's formais, devem fornecer apoio para: verificação e análise formais, simulação e execução.

2.4.1. Elementos de Processo

Um modelo apropriado de uma situação do mundo real consiste de uma descrição da estrutura e comportamento de todos os elementos do problema relevante (isto é, objetos), bem como seus inter-relacionamentos estáticos e dinâmicos.

Um dado domínio de problema, como os de processos de software, possuem elementos e relacionamentos que podem ser classificados em elementos e relacionamentos constituintes do domínio (Conradi e Jaccheri, 1998). Um modelo de processo concreto consiste de instâncias desses tipos de relacionamentos e elementos (Conradi e Jaccheri, 1998) (Christie 1993). Um processo de software consiste de atividades de cooperação concorrentes, que abrange atividades de manutenção e desenvolvimento de software, bem como atividades de segurança, de qualidade e de gerenciamento de projeto.

Identificam-se os seguintes tipos de elementos básicos (ou primários) de um processo de software:

1. **Atividade**¹ - Um passo de processo concorrente, funcionando sobre artefatos de software e unidos à um papel humano que utilizam ferramentas de produção externa. Isto inclui canais de comunicação e diretivas de controle. Podem estar em diferentes níveis de abstração, isto é atividades podem ser decompostas. Atividades são o coração de qualquer processo.

Citam-se atividades de manutenção e desenvolvimento de software, bem como atividades de segurança, de qualidade e de gerenciamento de projeto. As atividades podem estar em qualquer nível granularidade e são geralmente associadas a papéis que podem ser assumidos por certas ferramentas e/ou usuários. Os artefatos constituem os operandos (entradas/saídas) de atividades. Desse modo, o núcleo de uma PML deve conter uma Linguagem de Manipulação de Dados (DML) para acessar tais artefatos.

2. **Produto** - Um artefato, persistente e versionado, simples ou composto, com relacionamentos mútuos. Os artefatos descrevem o produto na questão: produtos de software e compostos deles, documentos associados, por exemplo documentos do projeto, documentação de usuário e dados de teste. Em um sistema reflexivo, todos os fragmentos do modelo do processo poderão ser considerados artefatos, isto pode incluir; meta-dados, os próprios modelos de processo e manuais de qualidade. O modelo do produto conterá normalmente um modelo de dados básico, um esquema de produto, e instâncias do produto. Pelo menos as dependências e composição do produto devem ser descritas.

¹ Algumas abordagens para modelagem de processos utilizam o termo Tarefa, considera-se neste trabalho atividade e tarefa como sinônimos.

3. **Papel** - Um papel descreve os direitos e responsabilidades da pessoa (humano) que estará encarregado de uma atividade.

4. **Pessoas (Humanos)** - equipes e agentes do processo onde: Um papel acima poderá ser assumido por pessoas (humanos) com capacidade para exercer este papel. Um usuário humano ou agente de processo poderá assumir um grupo de papéis. Ele também poderá ser um membro de várias equipes ou grupos, possivelmente representando equipes de projeto ou linhas organizacionais.

5. **Ferramentas** - para produção de software: Isto abrange ferramentas interativas e de lote como por exemplo: ferramentas CASE, editores de documento, e compiladores. O modelo de ferramenta deve especificar como as ferramentas poderão ser acessadas e controladas. Deve-se distinguir entre as ferramentas interativas e as ferramentas de lote. As ferramentas de lote abrangem compiladores, parsers, etc. As ferramentas interativas incluem: editores de documentos e ferramentas CASE.

6. **Suporte à Evolução** - O meta-processo geral suporta variabilidade dinâmica e estática de modelo de processo. Devido à natureza orientada a humanos do processo de software, procura-se evoluir o processo durante sua execução (*enactment*). Isto significa que a maioria das fases prévias do ciclo de vida deve ser repetidas *on-the-fly*. Assim o núcleo da PML (core) deve oferecer suporte para evolução de modelos de processo, tanto tecnicamente (por exemplo reflexão ou interpretação) como conceitualmente (por um meta modelo definido).

Um modelo de processo consiste de instâncias desses elementos de processo juntamente com as restrições adicionais (constraints) com que eles podem se inter-relacionar. Estes tipos de inter-relacionamentos são mencionados implicitamente nos parágrafos anteriores, por exemplo, uma atividade *produz* artefatos ou agentes *desempenham* um papel.

Esses elementos básicos e relacionamentos permitem uma modelagem refinada de um processo de software. Uma linguagem de modelagem de processo de software tem que fornecer recursos de linguagem para modelar esses elementos básicos e avançados (ou sub-modelos) bem como seus inter-relacionamentos.

Existem muitas variantes sobre como uma PML apóia a modelagem desses ingredientes em um modelo de processo. Como não é possível abranger todas as características dentro de uma linguagem de especificação na mesma profundidade, o projetista (*designer*) da linguagem de modelagem do processo tem que tomar uma decisão em caso de “trade-offs”.

2.4.2.- Fases de Modelagem e Automação de um Processo

Um modelo de processo é usado de diferentes maneiras por diferentes tipos de usuários durante diferentes fases do meta-processo (Conradi e Jaccheri, 1998). Nesta seção, menciona-se as principais fases do meta-processo juntamente com seus requisitos específicos.

1. Especificações de Requisitos e Extração do Processo “Elicitation” -

Durante esta fase, o modelador do processo necessita configurar metas gerais, e entender as ramificações destas metas, através de um modelo de processo abstrato. Nessa fase, deve-se questionar e negociar, procurando o entendimento humano sobre como o processo é entendido (percebido). Torna-se necessário, fazer a modelagem geral (conceitual), isto é: explicar as regras de negócios, fluxo de trabalho, e responsabilidades gerais de trabalho.

As freqüentes anotações gráficas e intuitivas podem ser importantes se a audiência for de formadores de opinião (decisão) da empresa. A PML nesta fase, deve se assemelhar às linguagens usadas para Sistemas de Informação e Modelagem de Negócios (empreendimento).

2. Análise do Processo - Nesta fase, a PML deve ser suficientemente formal para ser usada para raciocínio ou simulação. As necessidades alteradas a partir dos mercados e novas introduções de fornecedores de tecnologia podem entrar nesta fase. As decisões nas alterações para definir um processo normalmente se darão nesta fase, deste modo, questões de desempenho e qualidade devem ser lidadas.

Os Analistas de Processo têm de fornecer um modelo compreensível, consistente e verificável da organização do processo.

3. Projeto do processo - Nesta fase, a PML deve ser capaz de expressar uma arquitetura de processo mais detalhado e incorporar mais informações específicas do projeto como, por exemplo número de sub projetos, planejamento geral (dependências, cronometragem), tecnologia de desenvolvimento (técnicas OO), modelo de qualidade etc.

Os *designers* de processo precisam descrever como o modelo de processo tem de ser implementado em um PSEE. O papel do designer e o analista do processo é freqüentemente intercalado num papel de engenheiro de processo.

4. A implementação do processo – Nesta fase, a PML deve permitir especificação de detalhes de baixo nível que um modelo de processo interpretável deve alcançar. Este modelo deve possivelmente ser traduzido e preparado para execução.

Os **programadores do modelo do processo** devem codificar as decisões acima em uma PML executável.

5. Execução “Enaction” do processo – Deve-se, iniciar uma máquina de processos (discutido mais detalhadamente no Capítulo 3), deve ser capaz de executar o modelo de processo, que deve estar residente em um repositório de um PSEE. Esta máquina de processos interage com as ferramentas de produção e com os agentes de processo. Esta interação ocorre através de uma

interface com as ferramentas (por exemplo, um BMS) e através de uma interface de usuário (por exemplo através de uma agenda).

Os agentes do processo precisam de alguma visão, ainda que personalizada, para suas atividades específicas.

Os gerentes de projeto precisam de uma visualização de gerenciamento do projeto, uma vez que eles precisam entender, planejar, e controlar a estrutura do projeto e progresso do trabalho.

6. Avaliação do processo - isto inclui qualidade e desempenho, convergindo para qualquer coisa desde o trivial como ativações de ferramentas até coletas de métricas do processo. Todos os dados poderão ser fornecidos como feedback (informação de retorno) para as fases anteriores, tanto para guiar o processo ou possivelmente envolver o modelo de processo e seu processo. Um modelo de Desempenho e Qualidade deve regular tudo isso, e, as ferramentas de produção devem adequadamente ser utilizadas para este propósito.

As características bem conhecidas de quaisquer linguagens de programação são:

- **Formalidade** – Uma PML deve possuir sintaxe e semântica formalmente definidas. As PMLs formais suportam, por exemplo, analisar precisamente as propriedades definidas para um modelo;
- **Expressividade** – Denota a capacidade de uma PML de expressar todos os aspectos de um modelo de processo, que pode ser modelado diretamente utilizando recursos de linguagem da própria PML;
- **Gráfico** - Característica atribuída a linguagens que possuem elementos gráficos, que possibilitam uma melhor visão do modelo por meio da utilização de diagramas específicos. Está estritamente

relacionado à compreensibilidade dos modelos de processo que é independente dos possíveis usuários de um modelo de processo.

A PML pode oferecer modelagem em amplos conceitos, como por exemplo, conceitos de abstração e modularização, para estruturar um modelo de processo em sub-modelos que sejam conectados através de certos relacionamentos. Os conceitos de abstração podem suportar a definição de sub-modelos abstratos mais gerais que sejam personalizados dentro de um modelo de processo concreto. Uma PML, também pode ou não pode oferecer a possibilidade de distinguir entre modelos de processo (instanciados) específicos e genéricos (template).

São características de uma PML:

- **Executabilidade** - PML pode suportar a definição de modelos operacionais. Modelos operacionais são modelos capazes de serem executados. Deste modo, os modelos operacionais definem as regras de execução e são facilmente ordenados.
- **Analisabilidade**: O PML pode suportar a definição de modelos descritivos, por exemplo, expressões lógicas predicadas. Os modelos descritivos são analisáveis facilmente.

PML pode suportar diretamente a evolução de modelos de processo. A reflexão é uma exigência importante. Então, existem parameterizações, vínculos dinâmicos, persistência e versionamento.

- **Visões/perspectivas conceituais múltiplas** - A PML pode suportar a definição de visões (consistente) em certas perspectivas de um modelo de processo. Isto implica mecanismos para integrar diferentes visões em um modelo de processo em um modelo de processo geral comum.

No próximo capítulo, serão abordados aspectos referentes a automação de processos, dentre os assuntos explorados estão respectivamente: Ambientes

de Engenharia de Software Centrados em Processo, o Ambiente e-WebProject, Tecnologias para automação de processos como workflow e sistemas de agentes por, exemplo.

CAPÍTULO 3

AMBIENTES DE ENGENHARIA DE SOFTWARE CENTRADOS EM PROCESSO E AUTOMAÇÃO DE PROCESSOS

Neste capítulo são apresentadas, inicialmente, as características de ambientes de engenharia de software centrados em processo, ambientes estes considerados como meio tecnológico para o efetivo apoio a esforços de melhorias de processos de software. Posteriormente, detalha-se o ambiente e-WebProject, ambiente no qual se insere este trabalho. Por fim as possíveis tecnologias para automação de processos que podem servir como base para a construção do serviço proposto nesta dissertação de mestrado.

3.1.- Ambientes de Engenharia de Software Centrados em Processo (Process-Centered Software Engineering Environments – PSEE)

Na área de Engenharia de Software, há cada vez mais interesse em conceber e desenvolver ambientes de engenharia de software que tornem possível e explícita a especificação dos processos de software (Sant'Anna, 2000).

No princípio, ferramentas separadas, como os construtores e compiladores, foram desenvolvidas para automatizar aspectos mecânicos do desenvolvimento de software, como, por exemplo, manter o controle dos endereços de memória e associar aos nomes de variáveis. Outras ferramentas como depuradores foram desenvolvidos para apoiar as partes mais intelectuais do processo.

Antes, as ferramentas apoiavam as atividades simples, como codificação, mas gradualmente foram desenvolvidas ferramentas para também apoiar as atividades mais complexas como, por exemplo, a análise de requisitos e projeto. Sistemas de arquivos permitiram o armazenamento de produtos de trabalho dentro do ambiente computacional. Até a um tempo atrás, foram

desenvolvidas ferramentas para as atividades individuais do software isoladas das outras atividades.

Os ambientes integrados como: Smalltalk (Ingalls, 1978), Interlisp (Teitelman e Masinter, 1984), e Unix (Ritchie e Thompson, 1974), aumentaram o nível de suporte ao processo, permitindo as várias ferramentas suportar partes individuais do processo de forma a cooperar e compartilhar os produtos de trabalho. A ferramenta Unix Make (Feldman, 1979) é uma das ferramentas que permitem ao usuário definir e automatizar partes do processo de desenvolvimento de software. *Shell scripts* servem para o mesmo propósito (Bourne, 1978).

Para integração adequada de ferramentas, alguns ambientes como o PCTE (Boudier et al., 1988) suportam o armazenamento dos objetos, estruturas e tipos, melhor que os arquivos *flat*. Ambientes recentes baseados no Ambiente de Field (Reis, 1990), por exemplo, SoftBench da HP (Cagan, 1990), fornecem mecanismos para integrar ferramentas independentes em um ambiente (distribuído) e para as ferramentas se comunicarem umas com as outras. Esses ambientes mais tarde foram chamados de Ambientes de Engenharia de Software, pois fornecem suporte integrado de múltiplas ferramentas e fornecem suporte para todas as atividades de software.

Os Ambientes de Engenharia de Software Centrado em Processo, ou *Process centered Software Engineering Environment* - PSEE (Christie, 1993) (Ben-Shaul et. al., 1995) (Ambriola et al., 1997) (Fuggetta, 2000), representam uma classe de ambientes que consideram a engenharia de software como a execução de um processo definido explicitamente. A ênfase em um modelo explícito do processo de software permite estender a visão do desenvolvimento de software para abranger três diferentes tipos de atividades, freqüentemente realizadas por grupos separados dentro da organização de desenvolvimento de software. As três atividades são:

- **Engenharia de processo** - define e avalia modelos do processo de software;
- **Gerenciamento de projeto** - seleciona um modelo de processo produzido pela engenharia de processo e o adapta para um projeto particular e então monitora a execução deste processo;
- **Engenharia de software** - executa um processo específico através do gerenciamento do projeto.

Na Figura 3.1, são mostradas essas três atividades e tipos de suporte que um PSEE pode fornecer a cada uma delas. O modelo de processo é o conector dessas três atividades.

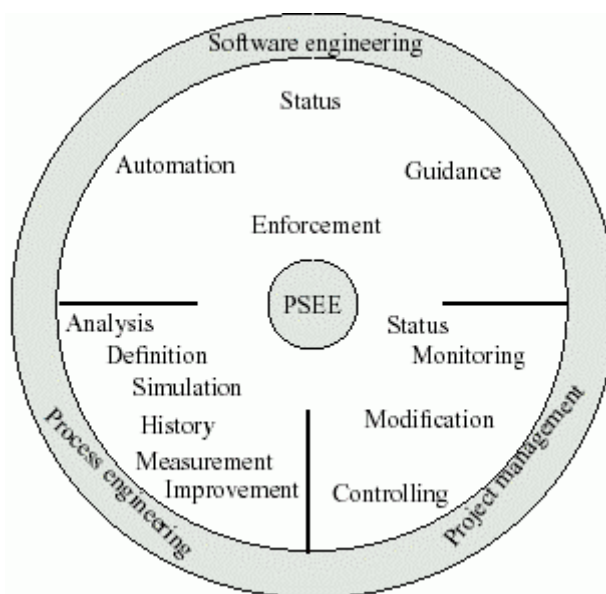


FIGURA 3.1.- O PSEE centraliza e integra o apoio para engenharia do processo, gerência de projetos e engenharia de software.

FONTE: Garg e Jazayeri (1996).

Para (Kadia, 1992), ambiente de engenharia de software pode ser definido como um conjunto de capacidades integradas para suportar os engenheiros de software e os gerentes em seu trabalho.

Na especificação SPICE (1995), esses ambientes são definidos como um conjunto integrado de ferramentas de desenvolvimento de software, para serem utilizadas pelos projetos na organização, suportando o processo de engenharia de software de forma consistente.

Sant'Anna (1993) define como um conjunto de ferramentas integradas a um banco de conhecimento de projetos para apoiar o desenvolvimento, a gerência e a experimentação de software.

Já em (Sant'Anna, 2000), o autor redefine ambiente de engenharia de software como um conjunto de recursos flexíveis e abrangentes, dando suporte eficiente e de forma integrada ao processo de engenharia de software definido para uma organização.

A tecnologia de processos de software propõe o desenvolvimento e adoção de PSEEs para automatizar a gerência dos processos (Gimenes, 1994). Esta tecnologia traz benefícios, como por exemplo: melhor comunicação entre as pessoas envolvidas e consistência do que está sendo feito; realização de algumas ações automáticas, permitindo que os seus usuários não realizem tarefas repetitivas; fornecimento de informações sobre o andamento do processo quando necessário; possibilidade de reutilização de processo de software e a coleta automática de métricas importantes para o controle e aperfeiçoamento de processos (Reis, 2001).

Ellmer (1995), menciona que os PSEEs serão a próxima geração de ferramentas CASE - *Computer Assisted Software Engineering*.

Os PSEEs estão se tornando uma realidade. Um crescente número de sistemas está revelando e aplicando processos para produção de software. Estes produtos e protótipos existentes são baseados em uma variedade de tecnologias e abordagens, como as linguagens e base de dados orientados a objetos, notações orientadas a estados, linguagens baseadas em regras ou mesmo linguagens lógicas (Ambriola et al., 1997).

Ambientes de engenharia centrados em processos se espalham nos ambientes de desenvolvimento de software de dois modos importantes:

- 1) Eles apóiam a definição e uso de um processo explícito de desenvolvimento de software. A forma, capacidade e tipo de apoio, variam entre os sistemas. No mínimo o ambiente deve apoiar a definição e o monitoramento do processo para detectar possíveis desvios e falhas no processo;
- 2) Eles vêem no processo de software mais do que uma descrição estática do ciclo de vida do software; eles vêem o processo como uma entidade dinâmica com seu próprio ciclo de vida. A primeira diferença entre um ambiente centrado em processo e os ambientes mais convencionais é o quanto eles apóiam as várias fases do ciclo de vida do processo de software. Realmente, PSEEs diferem de um para o outro, no nível de apoio que cada um provê nas diferentes fases do ciclo de vida do processo.

A capacidade de uma representação explícita de um processo é uma característica de definição de PSEEs. Um modelo de processo pode ser usado pelo PSEE para ajudar no projeto, análise, automação, monitoramento, e compartilhamento de processos de software. Vários PSEEs estão disponíveis agora tanto em comunidades acadêmicas quanto em comunidades industriais. Contudo, nem todos os PSEEs fornecem o mesmo conjunto de recursos.

As principais características dos PSEEs, são:

- **Definição de Processo** - Um engenheiro de processo usa um PSEE para definir um processo a ser seguido por um ou mais projetos. Os PSEEs atuais oferecem várias formas e critérios para especificar um processo. Algumas notações gráficas atuais dos PSEEs como ETVX, IDEF, diagrama de flechas, diagrama de estados, diagramas Entidade Relacionamento - ER, redes de Petri, ou diagramas de fluxo de dados.

Outros PSEEs permitem, ao engenheiro de processo, especificar o processo usando as tabelas ou formulários de texto. Outros PSEEs usam linguagem de programação para definir um processo. No paradigma da linguagem de programação, vários critérios foram testados, por exemplo, linguagens de *scripts* (extensões para linguagem de programação *Shell*), linguagens baseadas em regra (extensões para OPS-5 e Prolog), linguagens de base de dados ativadas por gatilhos (*trigger*) e linguagens de programação imperativa (extensões para ADA). Finalmente, vários PSEEs fornecem uma combinação de tais características algumas vezes até mesmo permitindo tradução de um para outro;

- **Análise de Processos** - Um modelo de processo dentro de um PSEE pode ser analisado em relação à consistência, integridade e corretude. Por exemplo, um modelo de processo pode ser analisado para determinar o número máximo de atividades paralelas possíveis em um processo (para propósitos de programação), ou para determinar se existem quaisquer atividades redundantes em um processo (as saídas que não estão sendo usadas). O processo pode ser analisado por conformidade com as diretrizes de qualidade que especificam um conjunto de atividades a serem realizadas em uma ordem particular, ou solicitar a existência de certas atividades de inspeção e revisão;

- **Apresentação do Processo** - Os processos de software descrevem fluxos de atividade que têm uma representação gráfica natural. Os produtos do processo também podem ser visualizados usando os diagramas estruturados. Um PSEE tipicamente suporta o *display* gráfico de informação de processo e produto. Tais displays gráficos são úteis tanto no papel quanto nos terminais eletrônicos. Normalmente, o *display* gráfico reflete o formalismo da definição do processo básico, por exemplo, uma definição baseada em redes de Petri poderá ser visualizada como uma Rede de Petri. Recentemente, vários PSEEs

estão defendendo a importância de fornecer visualizações múltiplas do processo, por exemplo, um gerenciamento comparado a uma visão de engenharia;

- **Simulação de Processo** - A execução de um processo de software normalmente exige várias pessoas, é realizada em um longo período e, poderá exigir um gasto monetário substancial. Para avaliar a conveniência de um processo antes de comprometer todos os recursos, alguns PSEEs suportam simulações de processo;

- **Automação do Processo** - Um processo de software freqüentemente inclui atividades que não exigem intervenção humana, este tipo de atividade pode ser automatizada. Um exemplo típico de uma atividade automatizável é executar testes de regressão em um módulo alterado recentemente. Notificar o pessoal afetado sobre uma alteração em alguma informação do produto é um outro exemplo. Uma vez que um processo foi definido em um PSEE, tais atividades podem ser identificadas (pelo engenheiro do processo) e automatizadas subseqüentemente pelo PSEE. O escopo e a extensão de tal automação depende da informação do processo e produto no PSEE, e o formalismo da definição do processo usado pelo PSEE. Por exemplo, se um PSEE não modela o pessoal envolvido em um processo, ele não poderá suportar efetivamente a notificação de alteração;

- **Monitoramento de Processo** - Um importante aspecto de um PSEE é monitorar a execução de um processo e registrar o histórico das atividades realizadas. Um histórico de processo pode ser usado para desenvolvimento e melhorias futuras do processo. Alguns PSEEs fornecem monitoramento em tempo real de um processo com *displays* gráficos, enquanto outros produzem dados do processo que podem ser analisados e visualizados com ferramentas de análise de dados. A privacidade de informação é um problema crítico aqui, uma vez que os

dados monitorados podem ser facilmente mal utilizados. Por outro lado, é geralmente aceito que as métricas de processo devem ser monitoradas para melhoria efetiva das práticas de engenharia de software;

- **Suporte a Alteração de Processo** - Os processos de software se alteram por várias razões: a tecnologia de programação pode alterar, por exemplo, programação orientada a objeto no lugar da programação estruturada; outros paradigmas de processo podem ser adotados, tais como: modelo de ciclo de vida espiral; ou, ferramentas de suporte que podem ser desenvolvidas e alteradas, por exemplo, programação orientada a janelamento (*windows*) no lugar de terminais baseados em caractere. Uma organização usando um PSEE deve ser capaz de alterar suas definições de processo dentro do PSEE, freqüentemente sem interromper o funcionamento da organização;

- **Interoperabilidade** - Os PSEEs têm que operar em ambientes de computação onde eles possam ou não ter controle sobre todas as ferramentas de software que estejam disponíveis. Alguns PSEEs fornecem facilidades de importação e exportação para intercambiar dados do produto entre as ferramentas existentes e o PSEE. Outros PSEEs estão de acordo com os esquemas de notificação baseados em evento padrão para integrar com as ferramentas existentes;

- **Suporte Multi-Usuário** - Um PSEE tipicamente tem múltiplos usuários. No mínimo, pessoas com funções de engenheiro de processo, gerente e engenheiro de software usam o mesmo PSEE. Múltiplos engenheiros de software freqüentemente trabalham no mesmo projeto, conseqüentemente, acessando o mesmo PSEE. Um PSEE fornece suporte para uma equipe de pessoas trabalhando juntas em um processo. Um PSEE assegura que o acesso concorrente para dados de produto e processo seja sincronizado;

- **Guia de Processo** - Os engenheiros de software usam um PSEE para realizar vários passos do processo. Um PSEE freqüentemente orienta um engenheiro de software sobre os possíveis próximos passos, baseado no processo modelado e seu estado atual. Alguns PSEEs fornecem tal orientação apresentando ao usuário uma lista de tarefa dos itens de ação; outros apresentam ao usuário os objetos do produto atual sendo criados e as operações disponíveis neles; e ainda outros apresentam ao usuário um ambiente de programação tradicional com quase nenhuma orientação;
- **Interface de Usuário para Tarefa Específica** - Baseado em um processo modelado, um PSEE pode estender a interface usuário para reduzir o aumento da informação apresentada para o usuário. Ambientes de programação diferentes como Unix, que não têm qualquer entendimento do usuário para ser executado, um PSEE não tem que apresentar todas as ferramentas disponíveis de modo uniforme. Dependendo da tarefa à mão, um PSEE pode limitar as ferramentas que são aplicáveis para aquela tarefa. Semelhantemente, um PSEE poderá limitar os produtos visíveis ao seu usuário;

A explicação anterior apresenta uma união das características comumente encontradas em PSEEs de pesquisa e comercial existentes. Os PSEEs atuais suportam essas características e recursos em vários níveis.

Outros PSEEs, como o *Formalized System Development* (FSD) desenvolvido na *University of Southern California's Information Sciences Institute* (Balzer et al., 1983) foi um ambiente pioneiro com modelos explícitos de atividades de engenharia de software. O FSD fornece suporte à automação e ao monitoramento através do ambiente de engenharia de software.

Uma motivação semelhante guiou uma equipe de pesquisadores no Kestrel Institute para definir um ambiente de programação automática (Smith et al., 1985). Essas abordagens foram posteriormente combinadas em um projeto

chamado *Knowledge-based Software Assistant* (KBSA), com a meta de codificar formalmente o processo de software em um ambiente de engenharia de software para fornecer suporte automatizado para as diferentes atividades (Green et al., 1983). No mesmo período, dois outros projetos desenvolveram idéias semelhantes: além de “*intelligence*” no ambiente de programação Gandalf da Universidade Carnegie-Mellon (Kaiser e Feiler, 1987), e o desenvolvimento de um intérprete de programa de processo dentro do projeto Arcadia (Taylor et al., 1988). Baseados neste trabalho do começo dos anos 80, vários projetos foram iniciados no final dos anos 80 e início dos anos 90.

Algumas tendências gerais são evidentes nestes desenvolvimentos. O trabalho mais recente sobre PSEEs foi motivado para tornar os ambientes de engenharia de software mais “inteligentes”. Isto levou ao conceito de desenvolver modelos de processo formais e fazer códigos dentro do ambiente.

Subseqüentemente, os pesquisadores perceberam que o desenvolvimento do modelo do processo é uma atividade não-trivial. Isto tem levado vários sistemas a fornecerem suporte para desenvolvimento e análise de modelo de processo. Isto é uma tendência para os atuais PSEEs. Finalmente, pode-se indicar a tendência de que os PSEEs irão direcionar sua ênfase para suportar as contínuas melhorias de processo (CPI) para as organizações. Os atuais PSEEs não tratam este problema adequadamente, mas alguns trabalhos estão surgindo neste sentido.

No país, algumas propostas foram desenvolvidas, e dentre elas se destacam o ambiente ExpPSEE (Gimenes, 2000), a estação TABA (Oliveira, 1999) (Travassos, 1994), o APSEE-Prosoft (Reis, 1998) (Moraes, 1997) e o e-WebProject® (Sant’Anna, 2002).

O ExpPSEE permite definição precisa e instanciação de atividades do processo de software em relação a artefatos, atores e ferramentas utilizadas por atividades.

O projeto da Estação TABA, iniciou-se no início dos anos 90 e vem sendo desenvolvida pela Universidade Federal do Rio de Janeiro - UFRJ. A Estação TABA é um meta-ambiente capaz de gerar, através de instanciação, ambientes de desenvolvimento de software adequados às particularidades de processos de desenvolvimento de projetos específicos. Desta forma, a Estação TABA possui facilidades para a definição de processos, métodos e ferramentas CASE a serem utilizadas no processo de desenvolvimento. Esse ambiente foi inicialmente desenvolvido em Eiffel na plataforma UNIX e agora está desenvolvido em C++.

O APSEE-Prosoft está sendo desenvolvido no Instituto de Informática da Universidade Federal do Rio Grande do Sul - UFRGS, num programa de cooperação internacional com o *Institut fur Informatik - Universitat Stuttgart* (Alemanha). O objetivo principal deste ambiente é a gerência do processo de desenvolvimento de software. Sua última versão está sendo desenvolvida em Java.

O ambiente e-WebProject® será mais bem detalhado na sessão seguinte, pois este PSEE servirá de base para a definição e implementação da arquitetura que será proposta nas próximas sessões.

Souza et al. (2001), classifica e compara uma série de outros PSEEs, apresentando suas características e principais funcionalidades, em Derniame et al. (1999) são reunidos diversos aspectos mais recentes da tecnologia de processos de software.

3.2.- O Ambiente e-WebProject®

O PSEE e-WebProject® (Sant'Anna et al., 2002) é um produto comercial originado do trabalho de pesquisa "Um Ambiente Integrado para o Apoio ao Desenvolvimento e Gestão de Projetos de Software para Sistemas de Controle de Satélite - AMBGES" (Sant'Anna, 2000) desenvolvido no Instituto Nacional

de Pesquisas Espaciais - INPE. O principal objetivo do ambiente é fornecer um conjunto de ferramentas integradas para trabalho cooperativo, suporte à execução de processos de engenharia de software, gestão do conhecimento organizacional e apoio à infra-estrutura organizacional.

O e-WebProject é mais eficiente em empresas que procuram uma maior maturidade no desenvolvimento de software, enfim empresas que adotam modelos de qualidade como o CMM e normas ISO, já que o profissional que trabalha com ambientes deste tipo, deve ser disciplinado a trabalhos com processos definidos formalmente.

Este PSEE foi desenvolvido em Java e disponibilizado através da WEB/Internet, o que permite uma maior e mais fácil disseminação das informações.

3.2.1.Requisitos Definidos para o e-WebProject

Os principais requisitos definidos para este ambiente são

- Funcionais:
 - apoio ao trabalho cooperativo;
 - apoio à visibilidade dos produtos e dos processos;
 - apoio ao gerenciamento de forma efetiva;
 - suporte a projetos simultâneos;
 - melhoria dos passos;
 - flexibilidade na organização das equipes.
- Não funcionais:
 - Interoperabilidade;

- flexibilidade;
- extensibilidade;
- rapidez;
- eficiência;
- facilidade de uso;
- suporte à grande quantidade de usuários de classes diferentes.

3.2.2.Arquitetura Proposta

A arquitetura do ambiente, para atender às necessidades antes apresentadas, deve possuir as seguintes classes de componentes (Sant'Anna 2000):

- Elementos responsáveis pelo armazenamento e recuperação das informações de projeto, promovendo a integração da informação.
- Elementos responsáveis pela prestação de serviços.
- Elementos responsáveis pela interação do usuário com o ambiente.

Estes elementos são organizados de forma a refletir uma organização em camadas hierárquicas, como mostra a Figura 3.2.

- **Camada de armazenamento e recuperação:** responsável pelo armazenamento e recuperação das informações de projeto.
- **Camada de serviços:** vários elementos são necessários para que os indivíduos trabalhem de forma cooperativa. Para atender ao desenvolvimento de software, vários serviços podem ser prestados pelo ambiente de modo a atender os requisitos especificados. Estes serviços utilizam os elementos da camada de armazenamento e recuperação para obtenção das informações.

- **Camada de interação com o usuário:** o usuário deverá obter acesso ao ambiente a partir de qualquer ponto de uma rede de computadores. A utilização de navegadores (*browsers*) permite tal facilidade, haja vista a capacidade de operação em modo gráfico e em ambientes de janelas, possibilitando interação fácil e eficaz; além disso, existem navegadores para todas as plataformas existentes atualmente.

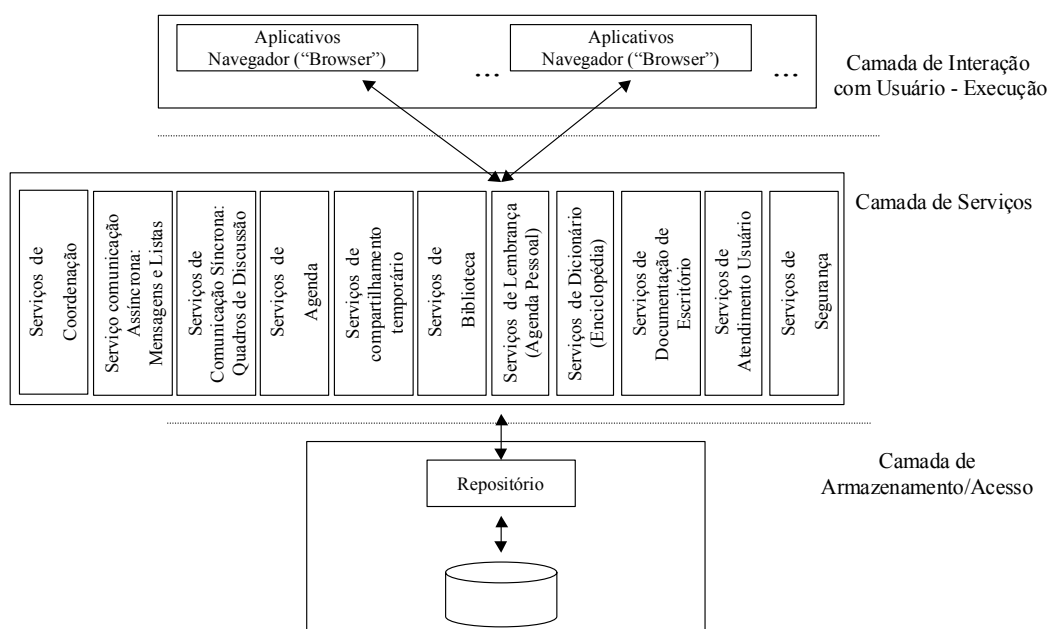


FIGURA 3.2.- Componentes e camada conceitual do ambiente.

FONTE: Sant'Anna (2000).

A arquitetura física é desenvolvida baseado no modelo "n-camadas" para aplicações Web (Conallen, 2000). A Figura 3.3 apresenta os componentes da arquitetura.

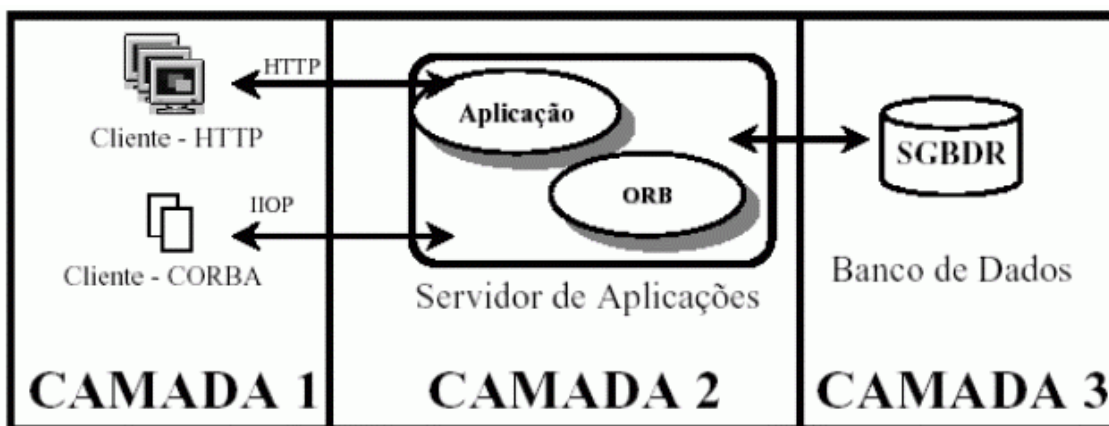


FIGURA 3.3.- Arquitetura física do ambiente.

FONTE: Sant'Anna (2000).

3.2.3.O Apoio à Execução de Processos no e-WebProject

3.2.3.1.O Processo: Ciclo de Vida

O processo é o elemento essencial do ambiente. Para que possa ser apoiado pelo ambiente, um modelo de processo necessita ser detalhado e aprovado pela organização, sendo transformado em um processo padrão. Portanto, considera-se neste trabalho que a automação de um processo somente será realizada após ter sido aceito para o projeto.

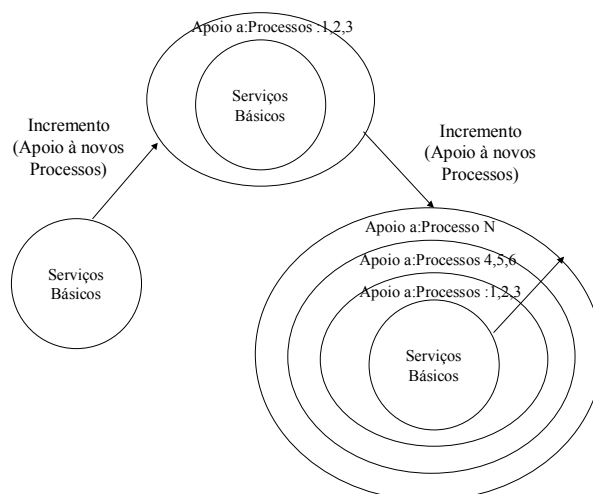


FIGURA 3.4.- O Caráter evolutivo do Ambiente.

FONTE (Sant'Anna, 2000).

Um processo padrão é instanciado no ambiente, permitindo que seja executado por pessoas e agentes autônomos. Durante a fase operacional do processo, ou seja, enquanto o ambiente estiver apoiando a execução do processo é possível realizar a coleta de dados (métricas) de eficiência, que após posterior análise, averigua-se a conformidade do processo padrão adotado, com os objetivos especificados. O modelo do processo é então melhorado e um novo processo padrão é adotado. A Figura 3.5 apresenta o ciclo de vida de um processo, tendo como fases:

- planejamento e a definição de um modelo de processo
- execução e acompanhamento de um processo
- avaliação e a proposta de melhorias do modelo de processo

3.2.3.2.- A fase de Planejamento de um Processo

Planejar um processo consiste de uma tarefa essencial para a automação de um processo. Um processo para ser automatizado necessita ter seu modelo bem entendido e aceito por toda a organização que o emprega. A fase de planejamento deve, portanto, permitir a especificação de vários elementos essenciais à execução do processo tais como, os recursos necessários à execução do processo, as entradas e saídas necessárias, quais as tarefas do processo e quem as executa.

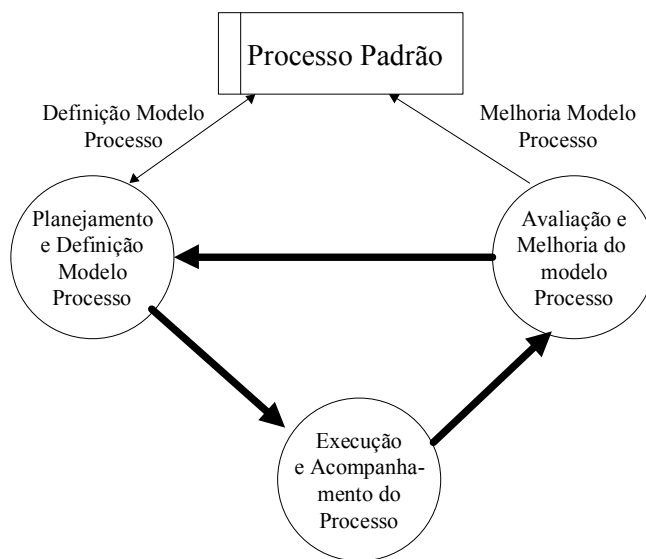


FIGURA 3.5.- Ciclo de vida de um processo.

FONTE Sant'Anna (2000).

Para este trabalho a fase de planejamento do processo deve passar por três etapas, a saber:

- preparação do modelo de processo
- aprovação do modelo pela organização
- fechamento do planejamento

A Figura 3.6 mostra as etapas de um planejamento do modelo de um processo.

A preparação do modelo de processos consiste inicialmente da descrição e detalhamento das atividades que compõem o processo no cotidiano real da organização. Estas atividades são descritas em termos de tarefas que são executadas pelos participantes. Além destas atividades iniciais a alocação dos indivíduos e dos recursos materiais às tarefas e a especificação de requisitos de entrada e saída de cada processo também são essenciais.

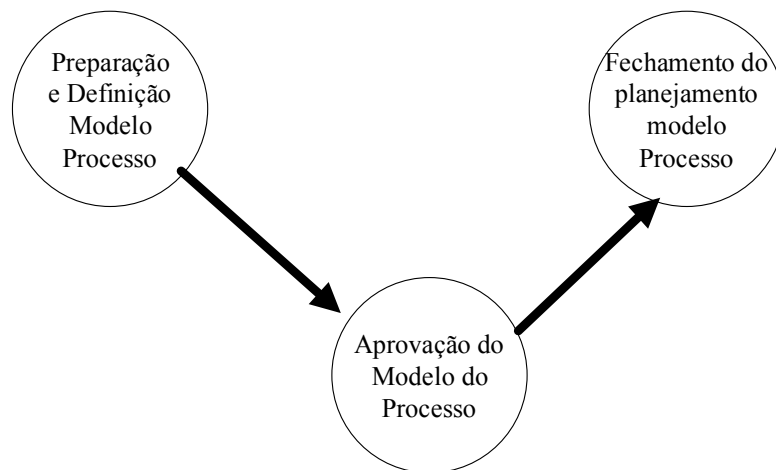


FIGURA 3.6.- A fase de planejamento.

FONTE: Sant'Anna (2000).

Apresenta-se a seguir uma seqüência de passos a serem seguidos, para a etapa de preparação do modelo de processo.

- Descrever a função e os objetivos básicos do processo a ser modelado
- Descrever os requisitos que o processo deverá atender.
- Identificar as entradas e saídas necessárias à execução do processo
- Definir os critérios para iniciar e fechar uma instância de processo
- Desdobrar o processo em atividades executáveis e factíveis descrevendo os procedimento para execução destas tarefas
- Relacionar as tarefas definidas mostrando a ordem de execução, e a dependência entre elas.
- Definir os papéis, descrevendo quais as características importantes para o executor de cada tarefa definida.
- Definir as responsabilidades, associando pessoas e agentes aos papéis definidos.

- Definir os recursos necessários
- Definir as métricas que são utilizadas para monitorar os processos.
- Definir os pontos de controle para melhor acompanhar a execução do processo.

A aprovação do modelo pela organização consiste de uma etapa de validação do modelo do processo definido na etapa de planejamento. Uma comissão de revisores deverá analisar e certificar o modelo definido. Uma vez o modelo certificado este se torna o modelo padrão do processo a ser adotada pelo ambiente e passa a ser disponibilizado para ser executado. Caso haja rejeição o modelo é reestruturado para posterior submissão.

3.2.3.3.- A Fase de Execução e Acompanhamento do Processo

O processo antes de tornar-se operacional no ambiente, precisa passar por uma etapa de preparação. Esta etapa de preparação consiste de uma série de passos a serem realizados, basicamente são atividades relacionadas à configuração do ambiente e ajuste para que o ambiente possa suportar o processo de modo planejado.

Uma vez concretizada esta etapa, o processo entra em um estado que é chamado de estado operacional (em operação). Poderá, então, quando neste estado, ser utilizado pelos participantes do projeto. A Figura 3.7 mostra os estados possíveis para um processo, quando em execução.

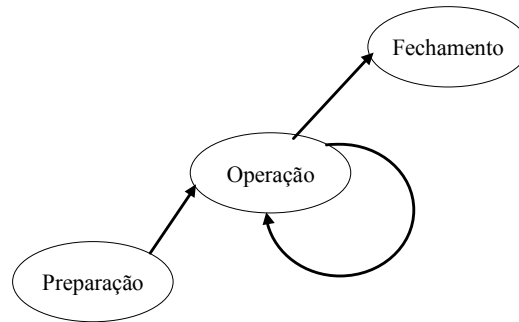


FIGURA 3.7.- Os estados possíveis para um processo quando na fase de execução.

FONTE: Sant'Anna (2000).

Quando for necessário parar de prover o serviço, o processo pode ser desativado e entra em um estado que é chamado de Fechamento.

Todo processo possui pelo menos uma atividade que o inicia. A execução desta atividade possibilita ao ambiente instanciar uma ocorrência do processo, e deve tomar as medidas necessárias para que o processo seja concluído conforme o especificado. Por exemplo, quando na identificação de um problema em um projeto, a atividade “registrar uma ocorrência” dispara um conjunto de ações que levam a solução do problema levantado. Este conjunto de outras atividades a serem executadas segue o modelo do processo proposto no planejamento. A etapa de operação permite, pois, que processos sejam instanciados e concluídos.

A execução de um processo implica que o ambiente deverá se preocupar com:

- Informar aos participantes de suas novas atividades a serem executadas e os prazos a serem cumpridos;
- Supervisionar as atividades, verificando se estão sendo cumpridas no prazo;
- Manter os gerentes e os participantes da execução do processo informado do seu andamento;

- Coletar as métricas de processo estabelecidas no planejamento para posterior avaliação;
- Tomar medidas de ação corretiva, em função das métricas de processo coletadas, para ajuste de pequenas inadequações.

O fechamento é a etapa onde o ambiente cessa de apoiar o processo. Para parar de prover este suporte, o ambiente deve ter a preocupação de:

- Informar aos gerentes e interessados que o ambiente deixa de prover o apoio ao determinado processo especificado
- Registrar em arquivos históricos, toda a informação referente ao trato do apoio ao processo especificado.

O término do suporte ao processo, normalmente ocorre por motivo de avaliação e melhoria do modelo padrão definido para o processo como será visto na seção a seguir.

3.2.3.4.- A Fase de Avaliação e Melhoria do Processo

A fase de avaliação compreende as atividades relacionadas com a análise das métricas de processo coletadas durante a fase de execução do processo. Esta investigação deve ser conduzida procurando identificar certas características, como por exemplo:

- Problemas na modelagem referente ao desdobramento do processo em atividades;
- Sobrecarga de atividade, ou seja, uma atividade muito executada causando falta de recurso;
- Atividades não necessárias;
- Pontos de estrangulamento causado por muitas restrições de entrada.

Após a análise das informações, medidas corretivas devem ser tomadas para que o apoio ao processo seja oferecido novamente. Um novo modelo deve ser elaborado e reconduzido para avaliação conforme descrito na fase de planejamento.

3.3.- Tecnologia de Workflow

Neste tópico é apresentada a tecnologia de fluxo de trabalho (*Workflow*) devido a sua similaridade com o proposto para a construção de PSEEs e uma referência genérica para a construção de máquinas de processos, elemento central de qualquer PSEE. Apresenta-se também o modelo de referência de sistemas do *Workflow Management Coalition*, que inspirou o desenvolvimento deste trabalho.

3.3.1.- Definições de Fluxo de Trabalho (Workflow)

O principal interesse de um fluxo de trabalho (*workflow*) é a automação de procedimentos onde documentos, informações ou tarefas são passadas entre participantes de acordo com um conjunto definido de regras para contribuir ou alcançar um objetivo. Um fluxo de trabalho pode ser organizado manualmente, mas, na prática, a maioria dos *workflow* são organizados no contexto de um sistema de TI para fornecer um suporte computadorizado para automação procedural.

O intuito da tecnologia de *workflow* está em possibilitar o trabalho integrado, cooperativo e ativo. Antes do surgimento das ferramentas que implementam estes conceitos, conhecidas genericamente como tecnologias para trabalho em grupo *groupware*, as atividades realizadas com computadores poderiam no máximo, servir como um complemento as atividades diárias.

A tecnologia de *workflow* possibilita a automação desses processos.

Existem na literatura algumas definições de *workflow*, dentre elas destacamos:

“A facilitação computadorizada ou automação de todo ou parte de um processo de negócio“.(Hollingsworth, 1995)

"*Workflow* é a tecnologia que possibilita automatizar processos, racionalizando-os e potencializando-os por meio de dois componentes implícitos: organização e tecnologia". (Cruz, 1998)

Workflow está freqüentemente associado com a reengenharia de processos de negócios, cujo interesse está na avaliação, análise, modelagem, definição e subsequente implementação operacional dos processos de negócio de uma organização. Implementações de *workflow* resultam de atividades de reengenharia de processos de negócios, mas nem todas estas atividades resultam em implementações. A tecnologia *workflow* é uma solução apropriada já que ela fornece uma separação da lógica dos procedimentos de negócio e seu apoio operacional de TI, possibilitando mudanças que podem ser incorporadas dentro de regras procedurais definindo o processo. Inversamente, nem todas as implementações de *workflow* formam parte de uma reengenharia de processo, como uma implementação para automatizar um processo de negócio existente por exemplo.

3.3.2.- Sistemas de Gerenciamento de Workflow

Um sistema de gerenciamento de *workflow* fornece automação procedural de processos de negócio através do gerenciamento da seqüência das atividades de trabalho e a invocação de uma pessoa apropriada e/ou recurso de TI associado com os vários passos das atividades.

O apoio a execução (*enactment*) de processos através de sistemas de gerenciamento de *workflow* tem sido utilizado substancialmente em domínios administrativos e técnicos. O desenvolvimento da tecnologia de *workflow* pode ser avaliado através de várias origens, tais como: sistemas de informação de escritório, trabalho cooperativo apoiado por computador (CSCW),

gerenciamento de documentos e imagens bem como as tecnologias avançadas de banco de dados.

Os sistemas de gerenciamento de *workflow* apóiam a execução dos processos coordenando a ordem lógica e temporal das atividades elementares de um processo (às vezes chamadas *workflows* elementares onde são descritos aspectos funcionais - fluxo de controle - e comportamentais de um *workflow*) fornecendo os dados, os recursos e os aplicativos necessários para a execução das funções (estes são os aspectos informativos, organizacionais e operacionais/tecnológicos de um *workflow*).

Um Sistema de Gerenciamento de *Workflow* pode ser definido como:

“Um sistema que define, gerencia e executa completamente fluxos de trabalho por meio de software cuja ordem de execução é guiada por uma representação computacional da lógica do *workflow*”. (Hollingsworth 1995)

Um processo de negócio individual pode ter um ciclo de vida variando de minutos a dias (eventualmente meses), dependendo de sua complexidade e duração das suas várias atividades . Alguns sistemas podem ser implementados de uma variedade de formas, usando uma variedade de infra-estruturas de comunicação e TI e operam em ambientes que variam de um pequeno grupo de trabalho local até operações complexas intra-empresas.

Embora haja a existência de toda esta variedade, todos os sistemas de Gerenciamento de *workflow* apresentam certas características comuns, que fornecem uma base para o desenvolvimento de um modelo de referência que possibilite a integração e interoperabilidade entre os diferentes produtos. O modelo de referência descreve um modelo comum para a construção de sistemas de *workflow* e identifica como as várias abordagens para implementação podem se relacionar.

Geralmente sistemas de *workflow* são a forma mais comum de automatizar processos orientados por pessoas como são os processos de software.

A importância do processo em um ambiente gerenciador de *workflow* se dá pelo fato de que o mesmo se torna o meio pelo qual a informação é processada.

No mais alto nível, todos os sistemas para gerenciamento de *workflow* podem ser caracterizados sob o ponto de vista de três áreas funcionais:

- Funções de Construção: responsável pela definição e possível modelagem dos processos de *workflow* e suas atividades constituintes.
- Funções de controle de execução: responsável pelo gerenciamento da execução dos processos de *workflow* em um ambiente operacional, fazendo o seqüenciamento das várias atividades a serem manipuladas como parte de cada processo.
- Interações em nível de execução com pessoas e aplicações de TI para processamento dos passos das várias atividades.

A Figura 3.8 ilustra as características básicas dos sistemas de *workflow* e os relacionamentos entre estas funções principais.

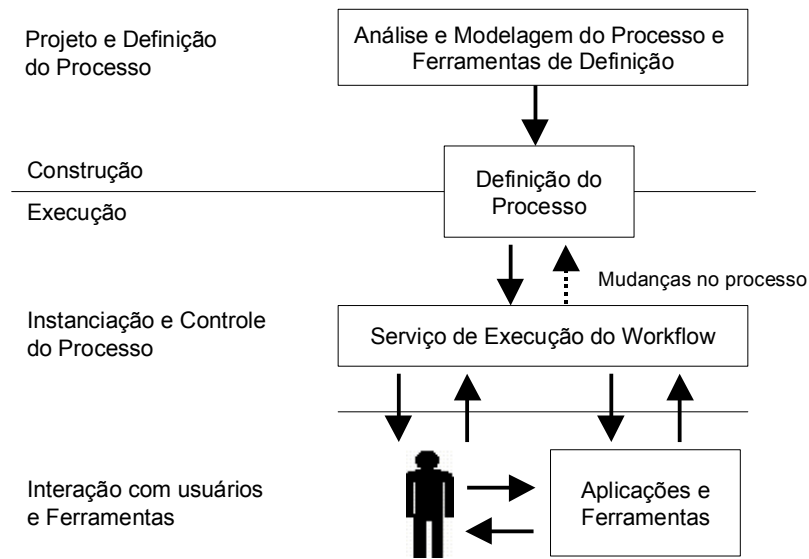


FIGURA 3.8.- Características básicas de um sistema de gerenciamento de workflow.

FONTE: adaptada de Hollingsworth (1995).

Sistemas de *workflow* podem ser construídos sob duas abordagens: baseado em eventos ou baseado em agentes autônomos.

Sistemas de *workflow* baseado em eventos são reativos, onde as ações são tomadas de acordo com os eventos ocorridos (finalização de tarefas, disponibilidade de recursos, etc). (Hagen e Alonso, 1999)

Sistemas de workflow baseados em agentes autônomos (Wang e Dongming, 2001) são mais flexíveis, agentes autônomos podem monitorar objetos de um processo (tarefas, recursos), e, além disso, tomar ações em resposta a eventos. Agentes podem assumir um papel pró-ativo, onde se pode, por exemplo, instanciar uma tarefa em paralelo sem que uma outra tenha sido finalizada, ou avisar a pessoa que está executando a tarefa que o tempo para finalização está atrasado.

3.3.3.- O Workflow Management Coalition (WFMC)

Muitas empresas, utilizam produtos de workflow, e há, uma oferta contínua de novos produtos no mercado. A disponibilidade de uma grande variedade de produtos no mercado permitiu a companhias individuais focar o desenvolvimento de seus produtos em funcionalidades particulares e usuários têm adotado produtos em particular para atender às suas necessidades mais específicas. Contudo, ainda não existem padrões definidos para que produtos de *Workflow* trabalhem de forma conjunta, no que resulta em “ilhas” de automação de processos incompatíveis.

O *WFM Coalition* é um agrupamento de companhias que têm trabalhado em conjunto para resolver a situação descrita acima. O WFMC têm reconhecido que todos os produtos de gerenciamento de *workflow* têm algumas características comuns, o que as habilita em potencial a alcançar um nível de interoperabilidade através do uso de padrões comuns para várias funções.

O *WFM Coalition* foi estabelecido para identificar estas áreas funcionais e desenvolver especificações apropriadas para implementação em produtos de workflow. Pretende-se que estas especificações irão habilitar a interoperabilidade entre produtos de *workflow* heterogêneos e irão melhorar a integração das aplicações de *workflow* com outros serviços de TI como correio eletrônico e gerenciamento de documentos, melhorando desse modo, as oportunidades para o efetivo uso da tecnologia de *workflow* com o mercado, beneficiando ambos usuários e desenvolvedores de produtos desta tecnologia.

3.3.4. Um Modelo de Referência para Implementação de Sistemas de Workflow

Em vista a variedade de produtos de *workflow* existentes no mercado, o WFMC desenvolveu um modelo de referência (Hollingsworth, 1995), as características descritas no modelo podem ser encontradas na maioria dos produtos, fornecendo uma base comum para a interoperabilidade entre os mesmos. Esta

abordagem identifica os principais componentes funcionais de um sistema de *workflow* e as interfaces entre eles, como um modelo abstrato.

Os componentes principais de um sistema de *workflow* genérico são ilustrados na Figura 3.9.

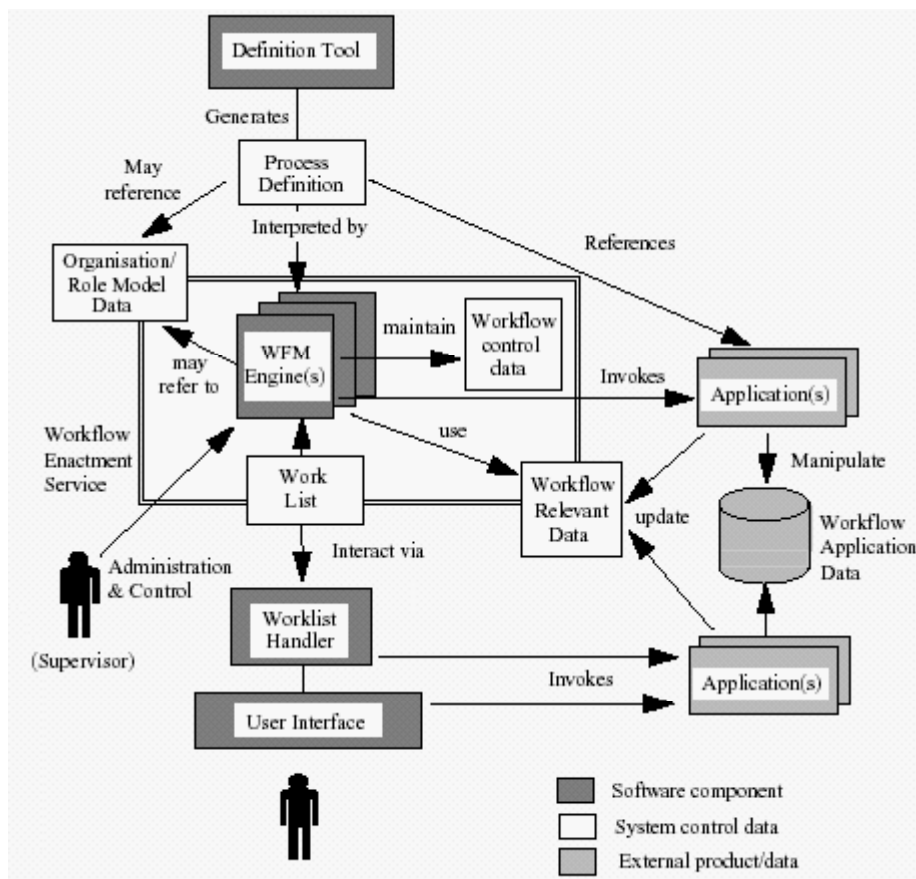


FIGURA 3.9.- Estrutura genérica de um produto de *workflow*.

FONTE: Hollingsworth (1995).

O modelo genérico possui três tipos de componentes:

Componentes de software que fornecem apoio às várias atividades do sistema de *workflow* (apresentadas em preenchimento mais escuro).

Vários tipos de definição sistema e dados de controle (apresentadas sem preenchimento) que são utilizadas pelos componentes de software.

Aplicações de TI e bancos de dados (apresentadas com preenchimento mais claro) que são parte do produto de *workflow*, mas que podem ser invocadas como parte do sistema de *workflow*

3.4.- Tecnologia de Agentes

Descreve-se nesta sessão característica de agentes de software, considerados elementos essenciais relacionados ao objetivo deste trabalho de pesquisa. Explica-se o conceito de agentes, uma visão panorâmica dos tipos de agentes existentes e, por último, uma descrição dos tipos de agentes considerados estratégicos para este trabalho.

3.4.1.- Definição de Agente

Tem-se tanta chance de aceitar uma definição consensual para o termo Agente como os pesquisadores têm para o termo Inteligência Artificial (IA). Existem ao menos, duas razões que provam o porque de toda esta dificuldade. Primeiro, pesquisadores da área de agentes não utilizam este termo do mesmo modo que pesquisadores que trabalham com outra área como a lógica *fuzzy*, por exemplo, o próprio termo lógica *fuzzy* é mais bem definido e difundido pelos pesquisadores da área, o que auxilia e muito no entendimento do significado do termo.

O mesmo não acontece com o termo agente que pode ter inúmeras variações tais como: agentes móveis, agentes de estado, etc. Segundo, a comunidade pesquisadora da área, define-se o termo agente, como um termo guarda-chuva que abrange um corpo heterogêneo de pesquisa e desenvolvimento.

A resposta de alguns pesquisadores para esta falta de um termo definido está na criação de alguns sinônimos, no intuito de minimizar esta imensa confusão citam-se como exemplos: *knowbots* (robôs baseados em conhecimento),

softbots (robôs de software), *taskbots* (robôs baseados em tarefas), robôs, agentes pessoais, agentes autônomos e assistentes pessoais. Existem algumas boas razões para estes sinônimos.

Primeiro, agentes atuam em ambientes heterogêneos, citam-se como exemplos: agentes que habitam o mundo físico, alguma fábrica, as vezes conhecidos como robôs; agentes que habitam em redes de computadores são conhecidos como *softbots*; agentes que realizam tarefas são conhecidos como *taskbots*; e, agentes autônomos referenciados por agentes móveis ou robôs que operam em ambientes dinâmicos e incertos. Segundo, agentes podem assumir muitos papéis, como assistentes pessoais ou *knowbots*, que possuem conhecimento específico em um dado domínio.

Considera-se para este trabalho de pesquisa, o termo agente como um componente de software que é capaz de atuar em ordem à cumprir tarefas para os quais foi projetado, de seus usuários, e que tenha capacidade de modificar o ambiente ao qual está inserido, como mostra a Figura 3.10. Dada essa definição para o termo, consideram-se as existências de alguns tipos de agentes, explicados adiante.



FIGURA 3.10.- Os agentes percebem as alterações de seu ambiente de agem de modo a modificá-los.

FONTE: adaptada de Weiss (1999).

3.4.2.- Tipologia de Agentes

Nesta seção é apresentada uma tipologia de agentes, que tem como intuito, referenciar o estudo de tipos de entidades que classificam alguns agentes de software existentes.

Primeiro, agentes de software podem ser classificados por sua mobilidade, isto é, sua capacidade de mover-se através de alguma rede de computadores. Esta classe se refere aos agentes móveis.

Segundo, agentes podem ser classificados por sua deliberatividade ou reatividade, isto é, agentes que possuem um simbolismo interno, um modelo racional no qual é engajado ao planejamento e negociação, em ordem a obter coordenação com outros agentes. Estes agentes não possuem qualquer modelo interno ou simbólico do ambiente ao qual ele habita ou atua, este tipo de agente atua em função de tipos de estímulos de comportamento, respondendo ao estado presente do ambiente de atuação.

Terceiro, agentes podem ser classificados através de vários ideais e atributos primários que devem possuir. Identifica-se uma lista mínima de três: autonomia, aprendizado e cooperação. Autonomia segue o princípio que agentes podem operar por conta própria, com pouca ou sem a necessidade de intervenção humana. Agentes possuem estados internos e objetivos individuais, desta maneira, os agentes agem de modo a satisfazer seus próprios objetivos e de seus usuários.

Um elemento chave para a autonomia é a pró-atividade, isto é, a habilidade de agir em resposta ao seu ambiente. Cooperação com outros agentes é a razão de ter múltiplos agentes em um ambiente em contraste de se ter apenas um. Para cooperar, um agente precisa possuir uma habilidade social, isto é, a habilidade de interagir com um outro agente, possivelmente humano através de alguma linguagem de comunicação.

Descritas estas características, uma questão surge. É possível para os agentes coordenar suas ações sem cooperação? Por último, para sistemas de agentes serem verdadeiramente inteligentes, eles devem aprender como reagir ou interagir com seu ambiente externo.

Para qualquer efeito, o termo agente deve ser desvinculado do termo inteligência. Neste trabalho, não é definido o termo inteligência, mas menciona-se uma característica essencial para algo que possua inteligência que é a capacidade de aprender. O aprendizado também deve contribuir para o aumento do desempenho ao longo do tempo.

A Figura 3.11 utiliza essas três características mínimas: cooperação, autonomia e aprendizado para derivar quatro tipos de agentes: agentes colaborativos, agentes colaborativos aprendizes, agentes de interface e agentes inteligentes proativos (*smart agents*).

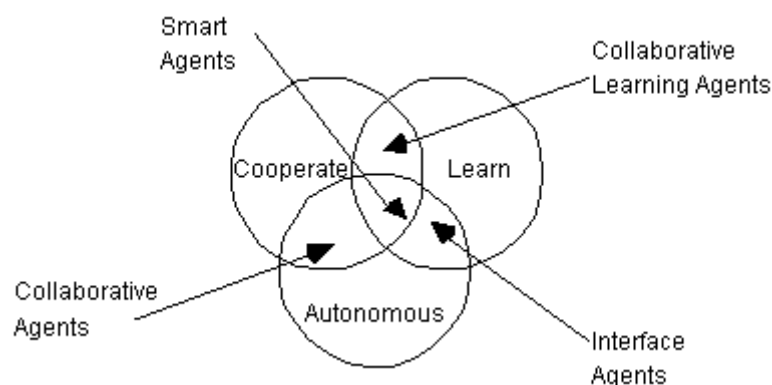


FIGURA 3.11.- Uma classificação de agentes de software.

FONTE: Nwana (1996).

Enfatiza-se que estas distinções não são definitivas. O que se tenta dizer é que a utilização de cada tipo de agente toma um determinado foco quando na construção de um sistema multi-agentes. Por exemplo, a utilização de agentes colaborativos tem mais ênfase na cooperação e autonomia do que no

aprendizado, mas isto não implica que agentes colaborativos nunca possam aprender.

Agentes de interface, no entanto, tomam um foco maior em autonomia e aprendizado do que em cooperação. Não foram encontrados relatos da utilização de agentes inteligentes pró-ativos, agentes estes, com capacidade de aprender com o ambiente em que atuam, podendo assim influir na melhora do desempenho do ambiente. Deste modo, procuramos, para este trabalho, desvincular a característica de inteligência dos agentes.

Por último, inclui-se a categoria de agentes híbridos que nada mais é do que a combinação das características dos tipos de agentes mencionados. Criando assim novos tipos de agentes como, por exemplo, agentes de interface colaborativos. Uma visão da tipologia de alguns agentes pode ser vista na Figura 3.12.



FIGURA 3.12.- Uma visão da tipologia de agentes.

FONTE: Nwana (1996).

Com todo este corpo de conhecimento relacionado aos agentes, onde se informam diversas características dos agentes tais como: mobilidade, colaboração, reatividade, proatividade, dentre outras. Identifica-se para este trabalho sete tipos de agentes:

- Agentes Colaborativos
- Agentes de Interface

- Agentes Móveis
- Agentes de Internet/Informação
- Agentes Reativos
- Agentes Híbridos
- Agentes Inteligentes Pró-ativos (smart agents)

Existem algumas aplicações que combinam agentes de duas ou mais categorias, referencia-se estas aplicações como sistemas heterogêneos de agentes.

3.4.3.- Uma Visão Panorâmica dos Diferentes Tipos de Agentes

O título deste trabalho de pesquisa é "Um Serviço de Coordenação de Processos Integrado ao Ambiente de Engenharia de Software e-WebProject". O próprio título sugere um serviço autônomo aplicado ao domínio da execução de processos de software, deste modo, considera-se a tecnologia de agentes como o melhor meio para chegar ao objetivo proposto com este trabalho. O próprio e-WebProject está estruturado em uma arquitetura web-based e centralizada, no qual foi projetada para atingir uma certa facilidade de acesso aos seus usuários. Para o escopo desse trabalho, consideram-se três tipos estratégicos de agentes: agentes colaborativos, agentes de interface, agentes reativos e agentes inteligentes. Tipos esses descritos a seguir.

3.4.3.1.- Agentes Colaborativos

Agentes colaborativos enfatizam autonomia e cooperação (com outros agentes) em ordem a realizar tarefas de seus donos. Eles podem aprender, mas este aspecto não é necessariamente a ênfase maior de sua operação. Agentes Colaborativos podem negociar de modo a alcançar aceitação mútua

com base em alguns padrões, possibilitando assim a coordenação de suas atividades.

As características gerais destes agentes incluem autonomia, habilidade social, responsabilidade e pró-atividade. Portanto, agentes colaborativos devem ser aptos a agir de forma racional e autônoma em ambientes multi-agentes abertos e restringidos por tempo. A Figura 3.13 apresenta a comunicação de uma colônia de cinco agentes distribuídas em duas máquinas distintas por meio de um facilitador (Nwana 1996).

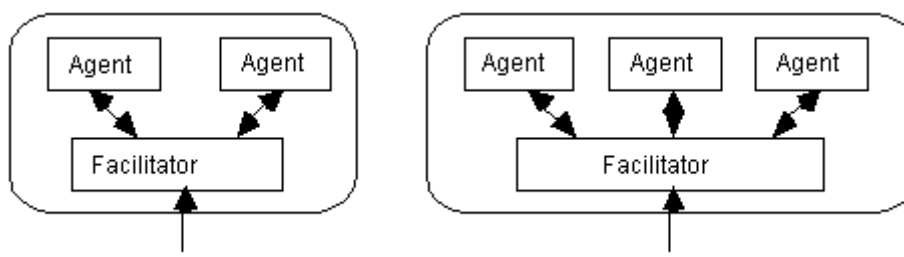


FIGURA 3.13.- Comunicação de uma colônia de 5 agentes.

FONTE: Nwana (1996).

3.4.3.2.- Agentes de Interface

Agentes de Interface enfatizam autonomia e aprendizado em ordem a realizar tarefas para seus donos. Como exemplo dessa classe de agentes, citam-se os assistentes pessoais que colaboram com o usuário no mesmo ambiente de trabalho. Observa-se a distinção entre a colaboração com o usuário e a colaboração com outros agentes como no caso dos agentes colaborativos. Colaboração com o usuário não precisa necessariamente de uma linguagem de comunicação de agentes explícita como no caso da colaboração com outros agentes.

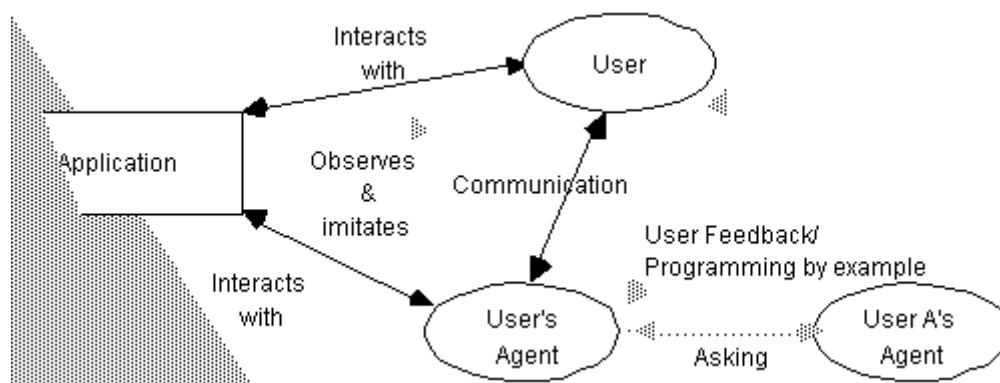


FIGURA 3.14.- Como Agentes de Interface Trabalham.

FONTE: Nwana (1996).

Essencialmente, agentes de interface apóiam e fornecem assistência ao usuário que está aprendendo a utilizar uma aplicação em particular como uma planilha de calculo ou sistema operacional. Os agentes de usuários observam e monitoram as ações do usuário com a interface, aprendem novos caminhos e sugerem um melhor modo de fazer a sua tarefa. Agentes de usuário atuam como um assistente pessoal autônomo que cooperam com o usuário de modo a completar alguma tarefa na aplicação. Agentes de interface aprendem a melhor assistir seus usuários de quatro formas distintas, as quais são apresentadas na Figura 3.14:

- Observando e imitando o usuário (aprendendo do usuário);
- Recebendo feedback negativo e positivo do usuário (aprendendo do usuário);
- Recebendo instruções explicitas do usuário (aprendendo do usuário);
- Perguntando a outros agentes para eliminar alguma dúvida (aprendendo de outros agentes).

A cooperação com outros agentes, quando existir, limita-se tipicamente a tirar dúvidas e não em efetuar negociação com eles como no caso dos agentes

colaborativos. Os modos de aprendizado são tipicamente por rotas (aprendizado baseado em memória) ou paramétrico, outras técnicas, como aprendizado evolucionário, também podem ser utilizadas.

3.4.3.3.- Agentes de Software Reativos

Agentes Reativos representam uma classe especial de agentes que não possuem um modelo interno ou simbólico de seus ambientes. Os agentes reativos agem/respondem a estímulos de maneira a afetar o estado do ambiente ao qual eles estão embarcados. O principal ponto de vista com respeito aos agentes reativos é que eles são relativamente simples e que eles interagem com outros agentes de forma básica. No entanto, padrões complexos de comportamento emergem destas interação quanto os agentes reativos são vistos de uma maneira global.

Três idéias-chave descrevem agentes reativos. Primeira, o dinamismo das interações torna a complexidade emergente, não existe uma especificação a priori (ou plano) do comportamento ou configuração dos agentes reativos. Segundo, decomposição de tarefas: um agente reativo é visto como uma coleção de módulos que operam de forma autônoma e que são responsáveis por tarefas específicas (monitorar, controlar, computar, etc). A comunicação entre os módulos é minimizada a uma natureza de baixo nível. .

3.4.3.4.- Agentes Inteligentes

Como mencionado, torna-se difícil considerar que um agente é inteligente. A principal questão levantada está em definir o próprio termo inteligência que não é uma tarefa trivial, que envolve muita pesquisa.

Para este trabalho, o termo agente inteligente, denota um agente de software com capacidade de efetuar ações de forma autônoma e flexível de modo a alcançar os objetivos para os quais o agente foi projetado (Weiss, 1999). Entende-se por flexibilidade a junção de três aspectos:

- **reatividade** - agentes inteligentes são habilitados a perceber o seu ambiente e responder em tempo de satisfazer seus objetivos de projeto;
- **pró-atividade** - agentes inteligentes são possuem um comportamento orientado a objetivos, o agente inteligente toma a iniciativa de modo a satisfazer seus objetivos de projeto;
- **habilidade social** - agentes inteligentes são capazes de interagir com outros agentes (possivelmente humanos) de modo a satisfazer seus objetivos de projeto.

Agentes inteligentes possuem estas três características e influem e alteram o ambiente onde estão inseridos. Neste trabalho, não existem agentes com características de pró-atividade.

3.4.4.- Perspectivas Tecnológicas para Implementação de Sistemas Multi-Agentes

Estão no contexto dessa seção: Uma breve descrição de uma arquitetura abstrata para software de agentes, uma breve descrição dos componentes da plataforma JADE, utilizado neste trabalho para a implementação do serviço de coordenação de processos.

3.4.4.1.- Especificação FIPA Abstract Architecture

A finalidade das especificações conduzidas pela *FIPA Foundation for Intelligent Physical Agents* está em promover a padronização, interoperabilidade e reusabilidade entre as várias implementações de plataformas para a construção de sistema multi-agentes. Para isto, torna-se necessário identificar elementos da arquitetura que devem ser codificados. Especificamente, se dois ou mais sistemas usam tecnologias diferentes para se obter o mesmo propósito funcional, torna-se necessário identificar características comuns relacionadas com as várias abordagens. Isto auxilia na identificação de abstrações

arquiteturais: projetos abstratos que podem ser formalmente relacionados a toda implementação válida.

Com respeito a especificação de arquitetura abstrata da FIPA, ela relata os elementos principais de uma arquitetura que deve ser utilizada para implementações concretas de sistemas de plataformas de agentes conforme mostra a Figura A.1. O foco primário da arquitetura abstrata está em criar um formato de intercâmbio de troca de mensagens entre agentes com um significado semântico, plataformas de agentes podem usar diferentes formas de transporte de mensagens, diferentes linguagens de comunicação de agentes e diferentes linguagens de conteúdo. Isto inclui vários pontos potenciais de interoperabilidade. O escopo da arquitetura abstrata, inclui:

- Um modelo de serviços e localização de serviços disponíveis para agentes e outros serviços;
- Interoperabilidade entre transporte de mensagens;
- Apoio às várias formas de representações de Linguagens de Comunicação de Agentes ACL;
- Apoio às várias formas de linguagem de conteúdo e;
- Apoio as múltiplas representações de serviços de catálogo.

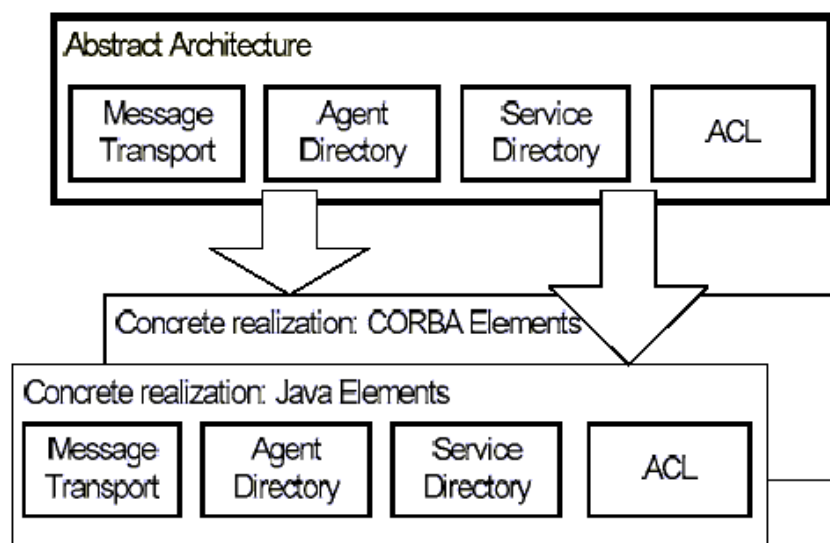


FIGURA 3.15.- Arquitetura abstrata FIPA mapeada para várias implementações concretas.

FONTE: FIPA (2002).

A arquitetura abstrata define quatro elementos básicos para prover a interoperabilidade entre a comunicação de agentes que são respectivamente: um serviço de transporte de mensagens, um serviço de catálogo de agentes, um serviço de catálogo de serviços e linguagens de comunicação de agentes. A FIPA também fornece especificações para outras necessidades referentes aos softwares de agentes tais como: ciclo de vida e gerenciamento de agentes, mobilidade de agentes, domínio, políticas conversacionais e identidade de agentes. Estas especificações podem ser encontradas em <http://www.fipa.org/>.

3.4.4.2.- A Plataforma de Agentes JADE

JADE (*Java Agent DEvelopment Framework*) é um *framework* de software totalmente implementado em linguagem Java. Este *framework* simplifica a implementação de sistemas multi-agentes através de um *middleware* que foi implementado em conformidade com as especificações FIPA e através de um conjunto de ferramentas que apóiam o debug e a fase de distribuição (*deployment*). A plataforma de agentes pode ser distribuída entre máquinas

(que não necessariamente precisam compartilhar o mesmo sistema operacional) e a configuração pode ser controlada via GUI remota. A configuração pode ser alterada em tempo de execução movendo os agentes de uma máquina para outra, quando solicitado. O único requisito de sistema solicitado é o *Java Run Time* versão 1.2.

A arquitetura de comunicação oferece um serviço de mensageria flexível e eficiente, onde a plataforma JADE cria e gerencia uma fila de recepção de mensagens ACL, privada a cada agente que pode acessar a fila através de uma combinação de várias formas: bloqueamento, pooling, timeout e baseada em satisfação de padrões. Todo o modelo de comunicação FIPA foi totalmente implementado e seus componentes foram distinguidos claramente e totalmente integrados: protocolos de interação, envelopes, ACL, linguagens de conteúdo, esquemas de codificação, ontologias e protocolos de transporte.

As plataformas JADE estão sendo utilizada por um número de empresas e grupos acadêmicos, ambos membros e não membros da organização FIPA, destacam-se CNET, Imperial College e Universidade de Helsink. A plataforma é disponível para uso sob licença GNU (open source) e pode ser encontrada em <http://jade.cselt.it/>.

No próximo capítulo será discutida a base conceitual do Serviço de Coordenação de Processos, onde são exploradas a utilização dos conceitos apresentados nos capítulos 2 e 3, serão apresentados todos os elementos que compõem o serviço em um nível mais alto de detalhamento.

CAPÍTULO 4

O SERVIÇO DE COORDENAÇÃO DE PROCESSOS DE SOFTWARE

Como visto nos capítulos 2 e 3, um modelo de processo deve ser automatizado por um ambiente de engenharia de software, para tanto, um serviço de coordenação de processos deve ser construído de modo a possibilitar: a definição ou edição de um processo de software; a execução do modelo de processo conforme ele foi especificado e definido para uma organização; guiar as pessoas envolvidas durante a execução do modelo de processo em um dado projeto; e, por fim, controlar os artefatos produzidos durante a execução do mesmo.

4.1.- Escopo do Serviço de Coordenação de Processos

Segundo Sant'Anna (2000), e conforme descrito no Capítulo 3, seção 3.2.3.1, o apoio aos processos no ambiente e-WebProject deve ser feito de forma evolutiva. Para ser apoiado pelo ambiente o processo passa por um ciclo de vida, que é composto por três fases:

- 1) Planejamento e definição de um modelo de processo;
- 2) Execução e acompanhamento de um processo;
- 3) Avaliação e proposta de melhorias do modelo de processo.

O serviço de coordenação de processos possui como escopo inicial para o contexto deste trabalho o apoio às fases 1 e 2. A sua principal atividade deve ser possibilitar pelo menos o apoio à definição de novos processos e à execução destes processos.

A fase 3, avaliação e proposta de melhorias de processos, levam em consideração outros aspectos como a coleta e análise de métricas de processos, bem como a utilização de técnicas de simulação para minimizar os

custos com a definição de novos processos e detecção de falhas. Estes aspectos levam ao estudo de outras técnicas, que possibilitam a detecção de falhas no processo, possibilitando a evolução dos processos, isso foge do escopo deste trabalho, que é dar o apoio à execução dos processos e deve ser discutido como um trabalho futuro.

Neste trabalho de pesquisa tem-se como intuito, modelar, especificar e construir pelo menos um serviço básico que possibilite, o apoio as principais fases de um processo de software e que permita uma posterior evolução.

4.2.- Requisitos para o Serviço de Coordenação de Processos

Conforme apresentado no Capítulo 2, um processo de software constitui-se de alguns elementos comuns (atividades, produtos, papéis, humanos, ferramentas), que cooperam entre si, de modo a produzir algum resultado.

Outra definição menciona que um processo se constitui de um conjunto de atividades parcialmente ordenadas para gerenciar, desenvolver ou manter sistemas de software (Acuña 2000).

Um sistema coordenador de processos tem como papel controlar as instâncias dos processos apoiados pelo ambiente.

Os principais requisitos para o serviço de coordenação de processos, propostos neste trabalho são

- Funcionais:
 - Possibilitar a definição de modelos de processos;
 - Ter a capacidade de executar os processos refletindo as restrições impostas no modelo;
 - Executar as tarefas que não necessitam da intervenção dos usuários;
 - Guiar as pessoas na execução das tarefas;

- Controlar os artefatos produzidos pelos processos;
- Monitorar os recursos utilizados pelos processos;
- Não funcionais:
 - Reatividade – o serviço deve perceber as alterações dos processos e tomar ações que viabilizem a sua execução.
 - Pró-atividade – o serviço pode tomar ações pró-ativas como, por exemplo, lembrar os usuários sobre tarefas que devem ser completadas e detectar falhas na execução dos processos;

Com base nessas características, especificou-se um serviço de coordenação de processos de software para o ambiente e-WebProject. Este serviço, encontra-se ainda em desenvolvimento, respeitando as características para o suporte evolutivo a processos de software definido para o ambiente.

4.3.- Uma Visão Geral do Serviço de Coordenação de Processos

O ambiente e-WebProject possui características para o apoio a todo o ciclo de vida de um processo de software. Um serviço de coordenação de processo para o ambiente em questão, deve seguir as premissas definidas por Sant'Anna (2000):

- 1) O ambiente de engenharia de software deve apoiar o processo de engenharia de software e seus processos componentes. Para tal, deve permitir que os processos sejam modelados, configurados, instanciados, acompanhados e terminados;
- 2) O serviço de coordenação dispõe dos meios pelos quais os processos são tratados pelo ambiente;
- 3) Como principal aspecto, este serviço deve garantir que um processo siga seu fluxo normal de execução dentro do ambiente;

- 4) O serviço deve avisar e relatar os participantes do processo em questão sobre o posicionamento do processo ou de ocorrências que o impossibilitam de terminar.

Com base nessas premissas, o serviço deve possuir facilidades para a configuração, execução e acompanhamento de processos. Para tanto foi definida uma arquitetura que possibilita a implementação destas facilidades. Foram levados em consideração também os requisitos descritos na seção 4.2.

Em um nível mais alto de detalhamento, foram definidos três sub-módulos principais, responsáveis por cada etapa do ciclo de vida de processo. Um sub-módulo responsável pela definição e/ou configuração de modelos de processo, outro responsável pela execução e acompanhamento do modelo, e o último, responsável pelo controle da produção durante a execução do modelo.

A arquitetura proposta para este serviço consiste de três elementos distintos que interagem entre si de modo a possibilitar um apoio efetivo a todo o ciclo de vida de um processo de software: desde sua definição até a sua execução. O serviço mencionado compõe-se de: um Ambiente para Definição de Processos de Software; Uma Máquina de Processos; e um Repositório de Artefatos. Os componentes do serviço para coordenação de processos encontra-se na Figura 4.1.

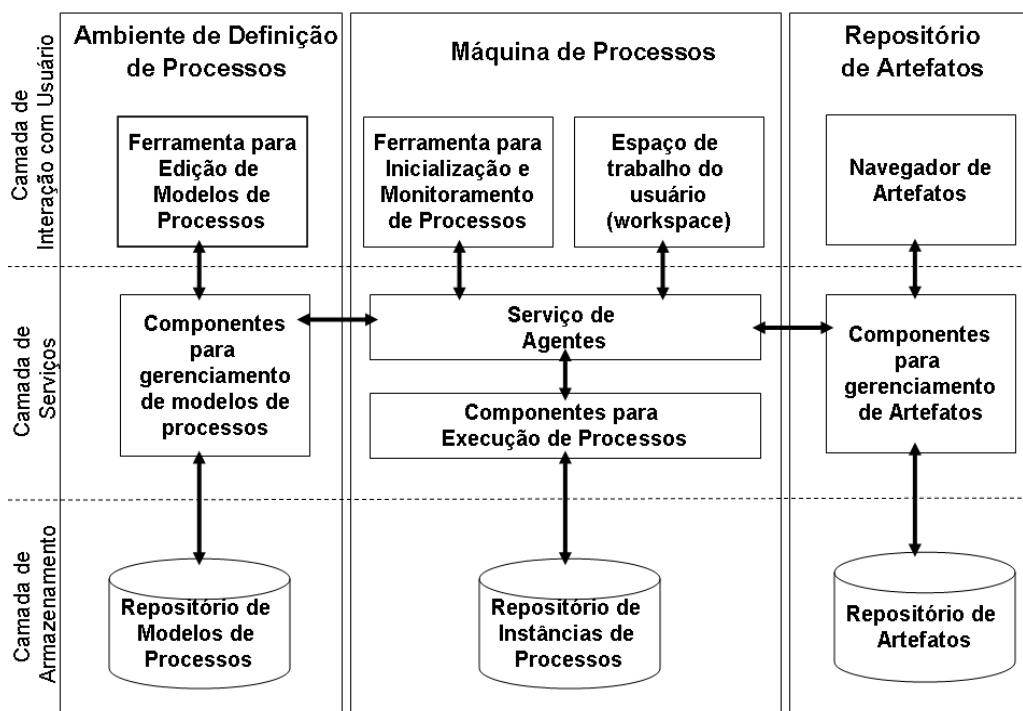


FIGURA 4.1.- Os componentes do Serviço de Coordenação de Processos.

Estes elementos, encontram-se constituídos por componentes que podem atuar em uma de três camadas distintas: a Camada de Interação com o Usuário, Camada de Serviços e Camada de Armazenamento, seguindo o modelo conceitual proposto para o ambiente e-WebProject, descrito no Capítulo 3.

Segundo Paula Filho (2000), um projeto representa um meio pelo qual se instanciam os processos. Nele, surgem participantes envolvidos em alguma fase do ciclo de vida do processo, participantes tais, que interagem com o serviço em uma dada fase do processo. Segue uma descrição de cada participante e sua forma de interação:

- **Modelador de processos** - pessoa responsável por descobrir as etapas de um processo, estruturar os elementos do processo em um modelo, e por fim, configurar este modelo no ambiente;

- **Gerente de Processos** - pessoa responsável por configurar e acompanhar uma instância de processo, durante a execução dos processos, o mesmo pode monitorar a execução das atividades, e pode configurar novas instâncias de processos, alocando pessoas para executar determinadas atividades;
- **Executor** - qualquer pessoa responsável por executar alguma tarefa, que produza algum tipo de artefato, ou seja, um elemento ligado diretamente com a execução do processo.

Note-se que este serviço necessita de uma certa “autonomia” para exercer as suas funções. Dados os requisitos definidos, no entanto, este serviço não pode apenas reagir em resposta a eventos, mas precisa também “perceber” eventuais problemas e tomar decisões pró-ativas, notificando os envolvidos quando necessário. A Tecnologia de Agentes é um meio para se conseguir resolver esses problemas. O Serviço de Agentes é o componente responsável por esta autonomia.

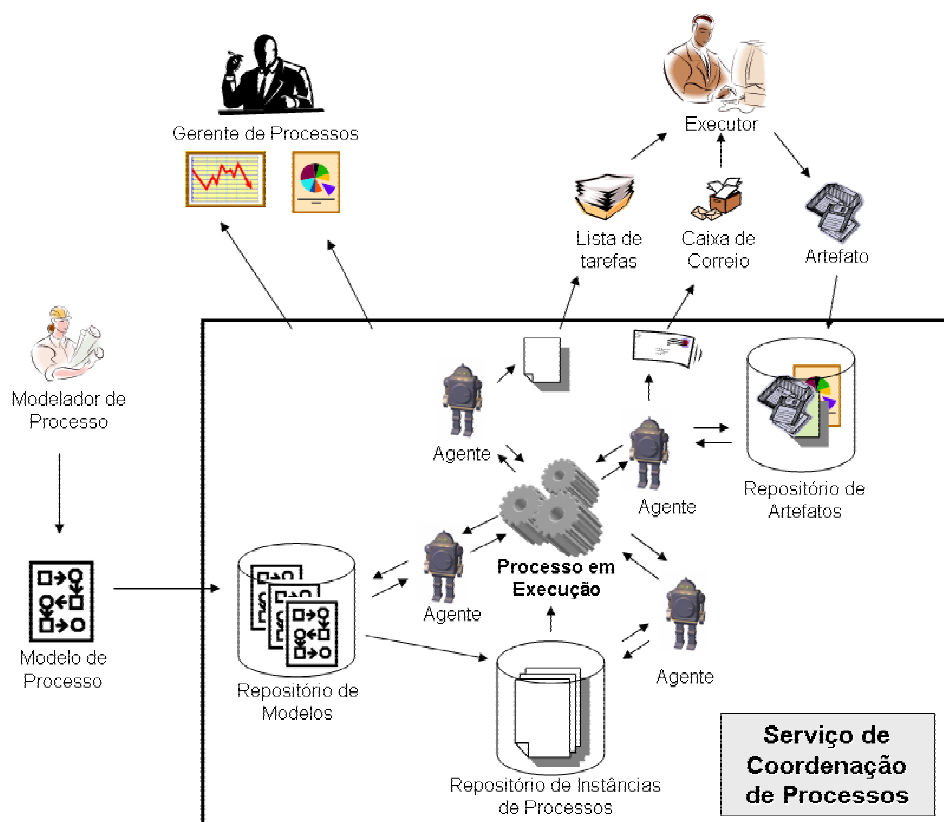


FIGURA 4.2.- A interação entre o serviço de coordenação de processos e seus participantes.

4.4.- O Ambiente de Definição de Processos

Este módulo ou ambiente (Cereja Jr et. al., 2003a), parte do serviço de coordenação de processos proposto, é responsável pela configuração de modelos de processo no ambiente e-WebProject.

4.4.1.- Os Componentes do Ambiente

Este ambiente, tem como principal característica, possibilitar da autoria e edição de modelos de processos, recurso importante para a padronização dos processos de software, item central em qualquer iniciativa de esforço de melhoria de processos (ISO, 1995).

O Ambiente de Definição de Processos batizado PDE (*“Process Definition Environment”*), foi desenvolvido com base em um meta-modelo que suporte o gerenciamento dos vários modelos de processos, este meta-modelo possui um conjunto de elementos comuns encontrados na maioria das técnicas de modelagem de processos atuais, deste modo, um modelador que tenha utilizado uma PML como a E3 (Jaccheri et. al., 1998) até mesmo a UML possa definir um modelo de processo de software com pouca ou nenhuma adaptação, de uma maneira relativamente simples.

A principal característica deste módulo está na possibilidade da autoria e edição de modelos de processos é parte da primeira fase da construção do serviço de coordenação de processos, um passo inicial rumo ao efetivo apoio a todo um ciclo de vida de um processo de software, incluindo seu monitoramento e sua execução, principal característica do ambiente e-WebProject.

A autoria de um processo pode ser feita utilizando qualquer uma das abordagens mencionadas no apêndice A. Depois de modelado o processo, seus elementos e associações são configurados no ambiente, por meio de interface gráfica. Um exemplo de utilização pode ser encontrado no Capítulo 9.

A Figura 4.3 apresenta a base conceitual do ambiente.

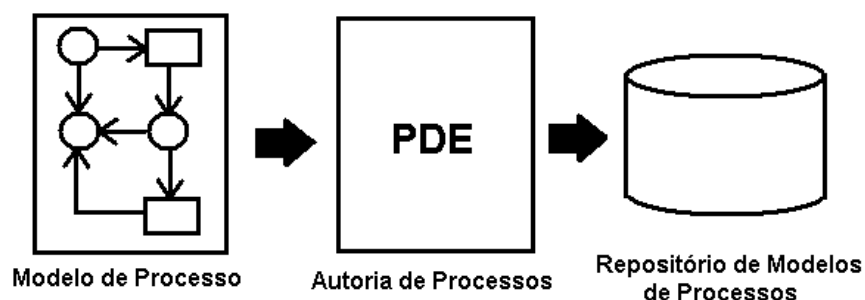


FIGURA 4.3.- Base conceitual da ferramenta PDE.

Este ambiente possui três elementos principais: um repositório de modelos de processos (camada de armazenamento), um conjunto de componentes capazes de manipular as informações destes modelos (camada de serviços) e um ambiente para definição e autoria de processos (camada de interação com o usuário). Uma descrição mais detalhada destes elementos vem a seguir:

- **Repositório de Modelos de Processos** - banco de dados construído com base em um meta-modelo de processos de software, responsável por armazenar as informações dos modelos, incluindo sua estrutura e regras de execução;
- **Componentes para Gerenciamento dos Modelos de Processos** - conjunto de classes de objetos persistentes responsáveis pela manipulação das informações contidas no repositório de modelos de processos;
- **Ambiente para definição e autoria de processos** - Ambiente gráfico de fácil utilização, capaz de auxiliar o modelador de processos a definir ou editar um modelo de processo que deve ser apoiado pelo ambiente.

4.5.- O Repositório de Artefatos

Este repositório, tem como principal finalidade o controle e gerenciamento dos produtos produzidos durante a execução de um processo de software. São considerados artefatos documentos, modelos, *templates*, elementos de modelos, enfim qualquer produto produzido por pessoas envolvidas na execução de um processo de software e que são parte do software que está sendo desenvolvido, isto inclui, planos de teste, código fonte, planos de projeto, relatórios de revisões, caso de uso, diagramas de classe, dentre outros tipos.

Quando discutido o controle e o gerenciamento de artefatos, surge o assunto “Gerenciamento de Configuração e Versão”, item imprescindível para controle dos produtos desenvolvidos em um dado projeto e mencionado como que

praticamente obrigatório em qualquer modelo de maturidade como o CMM (SEI 1993) por exemplo. Modelos para gerenciamento de configuração e versão foram desenvolvidos pela comunidade de engenharia de software dentre eles o Modelo Check in/ Check out se tornou o mais difundido e utilizado pela indústria de software.

O repositório de artefatos projetado para o serviço de coordenação de processos, possui facilidades para o gerenciamento dos produtos produzidos, onde foram especificadas funções para o check-in, check-out e congelamento (freeze) de versões produtos, além uma base de dados completa das meta-informações dos produtos. Informações como dados do artefato, dados de versões, tipos de artefatos, registro de modificações, etc. O repositório de artefatos possui três elementos que são respectivamente: O repositório de artefatos propriamente dito (camada de armazenamento), um conjunto de classes de objetos para manipulação das informações contidas no repositório (camada de serviços) e um navegador de artefatos *artifact browser* (camada de interação com usuário). Estes elementos serão descritos mais detalhadamente abaixo:

- **Repositório de Artefatos** - base de dados responsável pelo armazenamento dos produtos e suas respectivas meta-informações como dados do artefato, registro de versões, registro de modificações dentre outras;
- **Componentes para gerenciamento dos artefatos** - pacote classes de objetos persistentes capazes de manipular as informações e os artefatos armazenados no repositório, as funções para gerenciamento de configuração e versão estão implementadas nesta camada;
- **Navegador de Artefatos (*Artifact Browser*)**: Interface gráfica de fácil utilização que permite ao usuário, uma fácil recuperação dos arquivos

relacionados com os artefatos, além do acesso fácil as meta-informações do mesmo.

4.6.- A Máquina de Processos

É o coração do serviço de coordenação de processos, tem como principal finalidade controlar a execução do modelo de processo (*process enactment*) tal qual ele foi especificado, isto é, controlar a ordem da execução das atividades do processo, garantir que as restrições impostas ao modelo (uma tarefa deve produzir o artefato “relatório de revisão” e receber como entrada o artefato “diagrama de classes”, por exemplo) sejam respeitadas, a máquina de processos deve notificar as pessoas sobre a alocação de novas tarefas e deve monitorar as ações dos usuários, tomando ações para completar a execução dos processos.

A máquina de processos interage com o ambiente de definição de processos e o repositório de artefatos. A interação com o primeiro ocorre por causa da necessidade da interpretação dos modelos de processos definidos no ambiente que restringe a execução do processo. Neste caso a máquina de processos deve ser capaz de interpretar as restrições impostas ao modelo e guiar a execução do modelo respeitando estas regras.

A interação da máquina de processos com o segundo elemento, não menos importante, ocorre de forma mais branda. A máquina de processos consulta o repositório de artefatos para anexar um artefato de entrada a uma tarefa, ou para notificar o usuário que ele deve produzir um artefato de determinado tipo e que o mesmo deve estar no repositório após a finalização da tarefa em questão.

Para atender a todos estes requisitos, quatro elementos foram definidos para compor a máquina de processos que são respectivamente: Um repositório de instâncias de processos (camada de armazenamento), um serviço de agentes (camada de serviços), um espaço de trabalho *workspace* para usuário (camada

de interação com o usuário) e um ambiente para execução e monitoramento de instâncias de processos (camada de interação com o usuário). Uma descrição mais detalhada destes elementos é dada a seguir:

- **Repositório de Instâncias de Processos** - base de dados definida para manter as informações das instâncias dos modelos de processos, isto inclui as informações das atividades sendo executadas, pré-condições e pós-condições para a execução de uma tarefa, pessoas alocadas na execução das tarefas, dentre outras informações;
- **Serviço de Agentes** - elemento central da máquina de processos, consiste de um conjunto de agentes autônomos colaborativos capazes de diagnosticar a situação da execução dos modelos, planejar a execução das tarefas, monitorar ações das pessoas, notificar as pessoas dentre outras tarefas, este serviço de agentes foi definido dado a possibilidade dos agentes não somente terem um papel reativo durante a execução dos processos (executar a *tarefa 2* em resposta a finalização da *tarefa 1*, por exemplo), como também assumir um papel pró-ativo (informar ao gerente que uma determinada pessoa está super alocada, por exemplo), estes agentes analisam e manipulam as informações contidas nos repositórios e percebem os estímulos externos provenientes da interação dos usuários com as interfaces gráficas para realizarem seus trabalhos;
- **Componentes para Gerenciamento de Execução de Processos:** Componentes que representam as instâncias de Processos, são responsáveis por todo o gerenciamento dos elementos que compõem os processos e interagem com o Serviço de Agentes e o Repositório de Instâncias de Processos;
- **Espaço de Trabalho do Usuário (*Workspace*)** - ferramenta gráfica pelo qual o usuário recebe uma lista de tarefas que são alocadas ao

mesmo durante a execução do processo, por meio deste ambiente, o usuário também pode apontar tempos das tarefas, visualizar instruções, receber algum alerta do ambiente, além de possibilitar o acesso a algumas ferramentas colaborativas associadas ao ambiente e-WebProject. Enfim é uma área de trabalho eletrônica onde o usuário tem acesso a todo um conjunto de recursos que o auxiliam a realizar seu trabalho;

- **Ambiente para execução e monitoramento do processo:** ferramenta gráfica reservada aos gerentes de projeto que possibilita aos mesmos a inicialização de novas instâncias de processos e monitoramento do andamento da execução de processos já iniciados. Por meio deste ambiente, o gerente pode visualizar informações sobre a situação das tarefas do processo, monitorar os tempos de execução, visualizar informações que possibilitem a tomada de decisão durante o andamento do projeto.

4.7.- A Arquitetura Física

Nesta seção, descrevem-se os principais componentes físicos da arquitetura utilizada para a implementação do protótipo. Isso se torna necessário já que a arquitetura de desenvolvimento fornece meios para atender aos requisitos não funcionais propostos para o ambiente e-WebProject descritos no Capítulo 3, mais precisamente na seção 3.2.1.

Todos os componentes desta arquitetura foram especificados respeitando as características de aplicações WEB, característica central do e-WebProject.

Além dos requisitos não funcionais definidos no escopo mais geral do ambiente, foram definidos alguns requisitos adicionais de modo a atender as necessidades específicas do serviço de coordenação de processos proposto.

A tecnologia utilizada para implementação do e-WebProject é baseada na plataforma J2EE “Java 2 Enterprise Edition”, que é um padrão para a produção de aplicativos corporativos seguros, escaláveis e altamente disponíveis, além de outras características importantes como: flexibilidade, portabilidade, interoperabilidade, dentre outras necessárias à construção do e-WebProject. O padrão J2EE define um conjunto de serviços que devem ser fornecidos por servidores compatíveis com o padrão. Estes servidores fornecem contêineres responsáveis por abrigar os componentes de software J2EE.

Requisitos não funcionais definidos na seção 4.2, pró-atividade e reatividade, são atendidos pela implementação do Serviço de Agentes. Para implementar tal serviço, foi agregada a arquitetura do ambiente, a plataforma JADE, um componente adicional necessário à construção da máquina de processos, núcleo do serviço proposto.

Os elementos da arquitetura podem ser observados na Figura 4.4 no formato de um diagrama de componentes UML, uma descrição de cada elemento será dada a seguir.

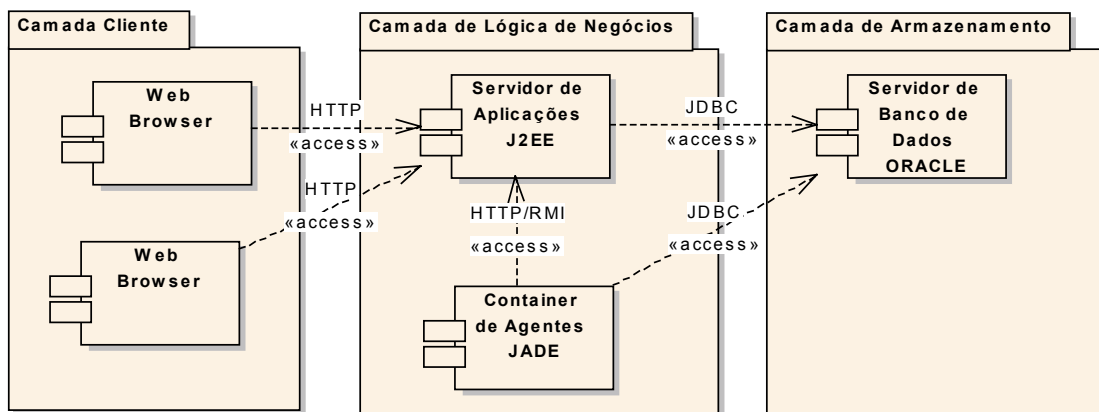


FIGURA 4.4.- Componentes da Arquitetura.

4.7.1.- Descrição dos Componentes da Arquitetura de Desenvolvimento

Cada elemento da arquitetura será descrito, primeiramente as camadas integrantes da arquitetura, posteriormente cada componente pertencente a camada descrita.

- **Camada Cliente** - Camada de acesso dos clientes a aplicação.
- **Web Browser** - Cliente leve (leve porque não existe processamento nesta camada). Envia solicitações a um servidor *Web* e recebe as respostas em formato HTML (*Hyper Text Markup Language*). Principais benefícios, facilidade de acesso e portabilidade, existem *browsers* para qualquer plataforma.
- **Camada de Lógica de Negócios** - Camada responsável pelo processamento da regras de negócio da aplicação, também conhecida como camada intermediária ou *middleware*, recebe as solicitações enviadas pelos clientes, efetua o processamento necessário e gera uma resposta adequada.
- **Servidor de Aplicações J2EE** - Servidor compatível com as especificações J2EE, deve possuir dois contêineres, um contêiner WEB e um contêiner de Componentes EJB *Enterprise Java Beans*, exemplos de servidores J2EE: *Jboss* e *IBM WebSphere*.
- **Contêiner WEB** - Contêiner responsável pela geração de conteúdo HTML dinâmico, os componentes de software principais deste contêiner são baseados na tecnologia JSP *Java Server Pages* e *Java Servlets*, que são componentes de software responsáveis por receber as solicitações de um *browser* cliente, fazer o devido processamento no lado servidor e gerar uma resposta dinâmica em formato HTML.

- **Contêiner de Componentes *Enterprise Java Beans*** - Este contêiner abriga componentes EJB. EJB é um padrão para desenvolvimento de componentes de software corporativos baseados na plataforma J2EE.
- **Contêiner de Agentes JADE “*Java Agent Development Framework*”**
 - Contêiner adicional agregado à arquitetura não possui nenhuma relação com o padrão J2EE. A Engenharia de Software Orientada a Agentes tem evoluído em larga escala nos últimos anos, tanto que foi formado um grupo que define um conjunto de especificações relacionadas a facilidades para o desenvolvimento de padrões para a construção de sistemas baseados em Agentes, este grupo denominado FIPA “*Foundation for Intelligent Physical Agents*” define este conjunto de especificações. O *framework* JADE é uma implementação destas facilidades baseadas nas especificações FIPA. Este *framework* foi escolhido por causa destas características e por ser desenvolvido totalmente em linguagem JAVA, sendo assim totalmente compatível com a arquitetura definida para o desenvolvimento do e-WebProject.
- **Camada de Armazenamento** - Camada responsável pelo armazenamento e recuperação dos dados processados pelo aplicativo, esta camada pode ser implementada por meio de várias formas, dentre elas: sistemas de arquivos, XML, Sistemas Gerenciadores de Bancos de Dados Relacionais e Orientados a Objetos, etc. Na arquitetura definida para o e-WebProject utilizamos um SGBDR Sistema Gerenciador de Banco de Dados Relacional.
- **Servidor de Banco de Dados ORACLE:** Servidor de banco de dados relacional. Servidor escolhido por atender bem aos requisitos de um SGBDR, tais como, recuperação contra paradas e falhas, controle de transações, múltiplos usuários simultâneos, segurança, integridade dos dados, confiabilidade e outros.

No próximo capítulo, são apresentados os aspectos de modelagem do Ambiente de Definição de Processos, primeiro módulo definido para o Serviço de Coordenação de Processos apresentado neste capítulo.

CAPÍTULO 5

O AMBIENTE DE DEFINIÇÃO DE PROCESSOS DE SOFTWARE

Neste capítulo é apresentada a modelagem de um ambiente para a definição de processos de software denominado PDE (Process Definition Environment) (Cereja Jr, M. G. et. al., 2003a). O ambiente PDE foi desenvolvido com base em elementos comuns, pertinentes ao domínio da modelagem de processos de software. Estes elementos, por sua vez, podem ser encontrados na maioria das técnicas de modelagem de processos descritas no Capítulo 2 e no apêndice A. Neste capítulo são descritas as fases de levantamento de funcionalidades e a análise do Ambiente de Definição de Processos.

5.1. Modelo Descritivo

Em linhas gerais é apresentado neste tópico o escopo do problema a ser solucionado com o desenvolvimento do Ambiente de Definição de Processos.

5.1.1.- Domínio do Negócio

Apoio à configuração e/ou definição de modelos de processos padronizados para uma organização.

5.1.2.- Descrição do Negócio

As Organizações que realizam atividades de desenvolvimento de software, para se adequar as exigências de qualidade necessárias ao sistema de software desenvolvido, devem se preocupar com a adoção de modelos de qualidade como o CMM, CMMI e SPICE. A formalização dos processos de produção de software de uma organização torna-se necessária para se alcançar o objetivo de se produzir software com qualidade, de uma maneira previsível.

Técnicas de modelagem de processos descritas no Capítulo 2, se apresentam como um meio para se efetivar a formalização de um processo no formato de

um modelo gráfico. No entanto esta atividade pode ser um pouco árdua ao modelador de processos, dependendo da complexidade do processo a ser modelado.

O modelador de processos, procura pelas atividades ou tarefas que compõem o processo, posteriormente, o modelador, para cada atividade, procura papéis responsáveis por sua execução, artefatos consumidos, artefatos produzidos e recursos necessários. Por último, o modelador organiza a ordem de execução das tarefas através da definição de relacionamentos. Isto se dá através da construção de um ou mais diagramas pertinentes a PML utilizada para a modelagem, representando assim a estrutura e comportamento do processo modelado. A Figura 5.1 apresenta um diagrama de atividades UML, representando as atividades de negócio modelar processos, descritas nesta seção.

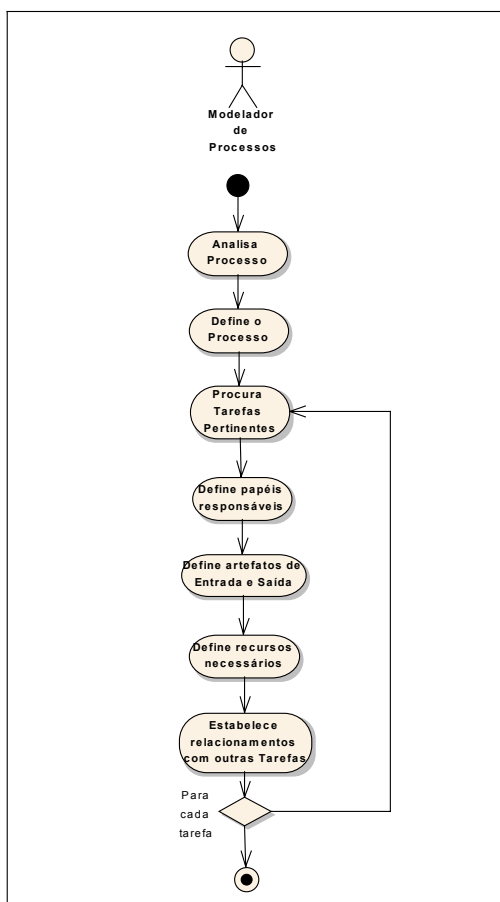


FIGURA 5.1.- Atividades do Domínio Modelar Processos.

5.2.- Modelo Conceitual

Nesta seção descrevem-se as funcionalidades do módulo de definição de processos de uma maneira mais detalhada, por meio da descrição de casos de uso.

5.2.1.- Objetivo do Sistema

Construir um sistema de software capaz de apoiar e/ou automatizar as atividades do negócio modelar processos.

5.2.2.- Atores e Funcionalidades

Atores:

Modelador de Processos: pessoa da organização que possui como principal responsabilidade a atividade de modelar processos, treinar as pessoas envolvidas e configurar o modelo no ambiente de engenharia de software possibilitando sua automação.

& suas funções

Modelador de Processos:

Mantém modelos de processos

Mantém Tarefas

Define papéis responsáveis

Define recursos necessários

Define tipos de artefatos de entrada

Define tipos de artefatos de saída

Define relacionamentos entre tarefas

5.2.3.- Visão Use Case: Contexto do Sistema

A Figura 5.2 mostra um diagrama de caso de uso, organizando as funcionalidades do sistema, relacionamentos e interação com os atores.

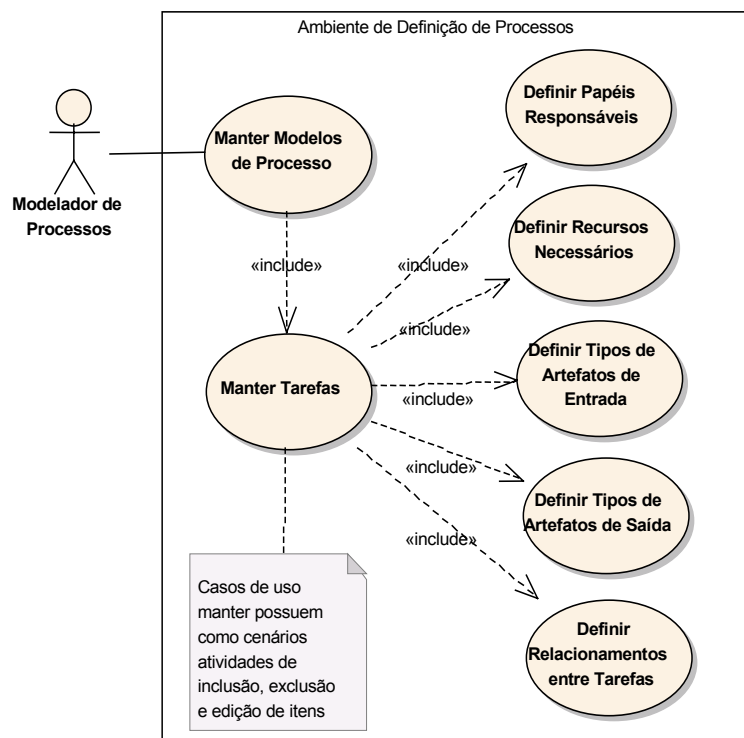


FIGURA 5.2.- Diagrama de Casos de Uso: Ambiente de Definição de Processos.

Na próxima seção, descreve-se de forma mais detalhada, cada caso de uso apresentado no diagrama que receberá um identificador no seguinte formato:

<UC>-<SIGLA DO MODULO>-<IDENTIFICADOR NUMERICO>

Exemplo: UC-PDE-01 onde, UC = abreviatura de use case e PDE = sigla adotada para o Ambiente de Definição de Processos.

5.2.4.- Descrição dos Casos de Uso

Segue abaixo a descrição do caso de uso Manter Modelos de Processo, uma descrição completa do conjunto de casos de uso pode ser encontrada no Apêndice C.

5.2.4.1.Caso de Uso: Manter Modelos de Processo

Identificador do Caso de Uso	UC-PDE-01
Nome do Caso de Uso	Manter Modelos de Processos
Atores	Modelador de Processos
Prioridade	Muito Alta
Pré Condição	Um usuário de perfil administrador válido deve ter sido autenticado pelo sistema
Requisitos Especiais	Não há
<i>Primeiro Cenário – Adicionar Modelo de Processo</i>	
Fluxo de Eventos:	
Fluxo Principal 1.- Este caso de uso inicia-se quando o modelador de processos seleciona a opção “New Process Model”; 2.- O sistema apresentará um formulário com informações referentes ao processo: Nome, Descrição, Objetivo, etc; 3.- O modelador de processos entra com os dados no formulário; 4.- O modelador de processos seleciona Insert; 5.- O sistema irá verificar e armazenar as informações.	
Fluxos Alternativos No passo 5, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário	
Pos Condição: O sistema apresentara uma mensagem de sucesso da operação.	
<i>Segundo Cenário – Editar Modelo de Processo</i>	
Fluxo de Eventos:	
Fluxo Principal 1.- Este caso de uso inicia-se quando o modelador de processos seleciona um processo na arvore de processos no lado esquerdo da Interface principal; 2.- O sistema busca as informações na base de dados e apresenta uma tela informando o nome do modelo e listando as tarefas que compõem o modelo; 3.- O modelador de processos seleciona “Edit Process Model”; 4.- O sistema apresenta um formulário com as informações atuais do modelo; 5.- O modelador de processos entra com as modificações; 6.- O modelador de processos seleciona “Update”; 7.- O sistema verifica e armazena as novas Informações;	
Fluxos Alternativos No passo 7, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro.	
Pós Condição:	

O sistema apresentara uma mensagem de sucesso da operação.
Terceiro Cenário – Excluir Modelo de Processo
Fluxo de Eventos:
Fluxo Principal 1.- Este caso de uso inicia-se quando o modelador de processos seleciona um processo na arvore de processos no lado esquerdo da Interface principal; 2.- O sistema busca as informações na base de dados e apresenta uma tela informando o nome do modelo e listando as tarefas que compõem o modelo; 3.- O modelador de processos seleciona “Delete Process Model”; 4.- O sistema apresenta uma mensagem para confirmação da exclusão; 5.- O modelador de processos seleciona Confirmar 6.- O sistema efetua a operação na base de dados;
Fluxos Alternativos No passo 4, se o modelador de processos selecionar “Não Excluir”, o sistema apresenta a interface descrita no passo 4. No passo 6, se ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao cliente
Pos Condição: O sistema apresentara uma mensagem de sucesso da operação.

5.3.- Modelo de Negócios

Nesta seção, inicia-se a análise e projeto das funcionalidades descritas na seção 5.2. Apresenta-se, em primeiro lugar, a modelagem das classes de negócio. Posteriormente, apresentam-se os diagramas de seqüência representando a interação (troca de mensagens) entre os objetos necessários à realização de cada caso de uso.

5.3.1. Modelo de Classes

A Figura 8.4 apresenta o diagrama de classes do repositório de modelos de processos, cada classe será descrita posteriormente.

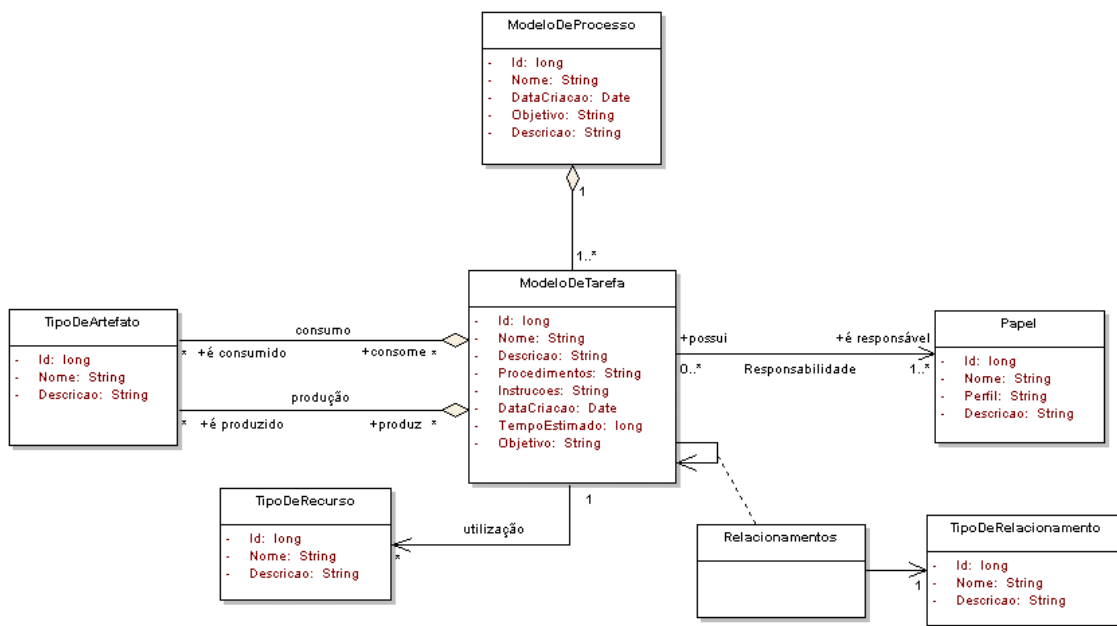


FIGURA 5.3.- Modelo de classes persistentes: repositório de modelos de processos.

A descrição de cada classe encontra-se na Tabela 5.1 apresentada abaixo:

TABELA 5.1.- Descrição das classes do repositório de modelos de processos.

Classe	Descrição	Instâncias
ModeloDeProcesso	Representa um processo formalizado pela organização	Processo de Solução de Problemas
ModeloDeTarefa	Representa uma Tarefa ou Atividade que compõe o Processo	Revisar Código Fonte
TipoDeArtefato	Representa tipos de artefatos que podem ser produzidos ou consumidos durante a execução de um processo	Caso de Uso, Documento de Requisitos, Plano de Teste
TipoDeRecurso	Qualquer recurso necessário para a execução da tarefa	Compilador, ferramenta case, Computador, scanner
Papel	Perfil necessário à execução da tarefa	Engenheiro de Requisitos, Analista de Sistemas, Revisor

(Continua)

Tabela 5.1 – Conclusão.

Relacionamento	Classe de Associação que representa os relacionamentos inter-tarefas do processo. Estes relacionamentos restringem a ordem de execução do processo	-
TipoDeRelacionamento	Representa alguns tipos de relacionamentos entre as tarefas	Precedência, sub-tarefa, feedback

Estas classes são a base para a manipulação dos modelos de processos, que devem ser configurados no ambiente, através delas o serviço de agentes pode estabelecer regras para controle de execução de tarefas, navegando entre os vários elementos de processo.

5.3.2. Modelagem da Interação Entre os Objetos

São apresentados nesta seção, os aspectos referentes à troca de mensagens entre os objetos necessários à realização de cada caso de uso definido na seção 5.2. A modelagem da interação dos objetos se dá por meio da utilização de diagramas de seqüência UML, definidos para cada cenário proposto ao caso de uso em questão.

5.3.2.1.- A Seqüência: Caso de Uso Manter Modelos de Processo

Apresenta-se, nesta seção, a seqüência de mensagens dos objetos que colaboram para realização do primeiro cenário Inserir Modelos de Processos, que pode ser vista na Figura 5.4. Um conjunto completo da modelagem da seqüência de mensagens para realização dos cenários dos casos de uso do Ambiente de Definição de Processos, encontra-se no apêndice C. Segue descrição da troca de mensagens abaixo:

- 1) O Modelador de Processos entra com os dados do processo e envia as solicitações ao ambiente;
- 2) O ambiente aciona o Componente ProcessarModelosDeProcessos para Processar a operação de inserção no banco de dados;

- 3) O Componente cria uma conexão com banco de dados em modo transacional, cria um Objeto ModeloDeProcessos, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado;
- 4) O componente solicita ao objeto ModeloDeProcessos para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeProcesso;
- 5) O objeto ProcessarModelosDeProcesso, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário.

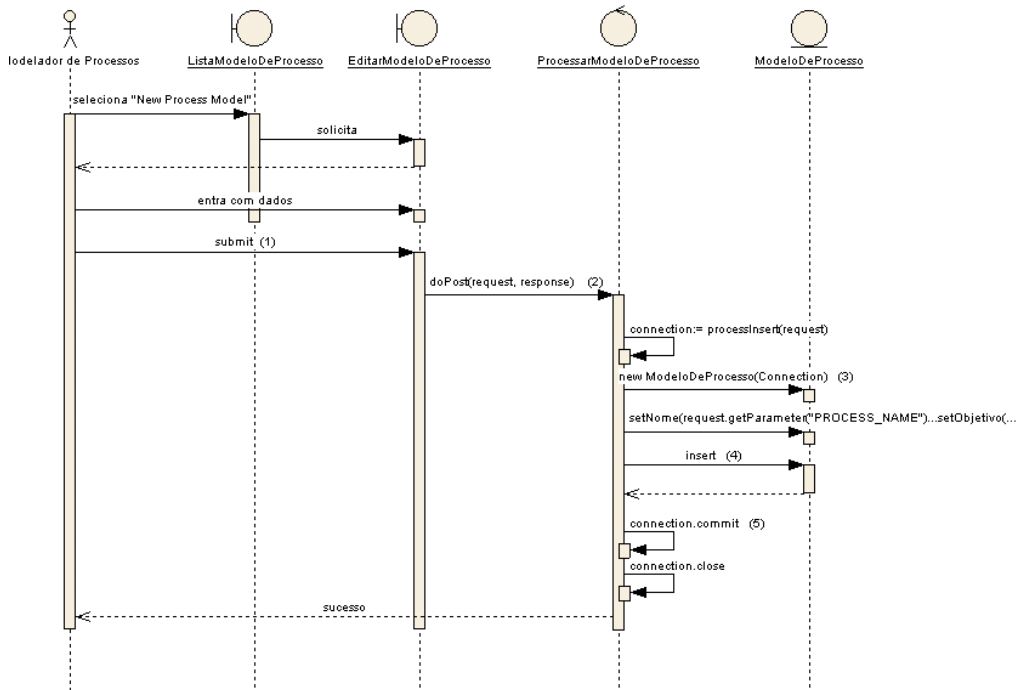


FIGURA 5.4.- Diagrama de seqüência: inserir um novo modelo de processo.

No próximo capítulo são apresentados os aspectos de modelagem da Máquina de Processos.

CAPÍTULO 6

A MÁQUINA DE PROCESSOS

Um processo após estar definido no ambiente, entra em fase de operação. Neste capítulo, descreve-se as fases de análise de requisitos e modelagem da máquina de processos, módulo do serviço de coordenação de processos responsável pelo controle de execução de instâncias de processos no e-WebProject.

Em primeiro lugar, define-se o escopo do problema a ser resolvido. Posteriormente, são descritos os casos de uso, serviços da máquina de processos e respectivas modelagem estrutural e comportamental.

6.1.- Modelo Descritivo

Descreve-se nesta seção, o escopo do problema a ser resolvido com o desenvolvimento da máquina de processos.

6.1.1.- Domínio do Negócio

Controle da execução de instâncias de modelos de processos de software.

6.1.2.- Descrição do Negócio

Modelos de processos descrevem a estrutura e o comportamento de um processo de software baseado em regras semânticas, estes modelos, por sua vez, conforme mencionado no Capítulo 2, podem ser utilizados para diversos propósitos, dentre eles podem ser citados: entendimento do processo, treinamento, simulação e suporte automatizado. Estes modelos podem ser armazenados em um repositório onde os elementos que compõem o processo são instâncias de entidades que representam meta-elementos pertinentes ao domínio da modelagem de processos, como modelos, tarefas, artefatos e relacionamentos, por exemplo, um repositório com estas características é apresentado no Capítulo 5.

Estas regras definidas nos modelos se tornam base para a execução das instâncias de processos, onde as mesmas regras, se tornam condições de disparo, finalização, planejamento e conclusão de elementos de instâncias de processos. Estas condições guiam a execução do modelo de processo em um dado projeto, fornecendo subsídios para a alocação de pessoas, detecção de causas de problemas, dentre outras.

Em primeiro lugar, é feita a leitura do modelo de processo, posteriormente é criada uma instância do modelo em um dado projeto, criando as condições necessários para a execução de acordo com o que foi definido no modelo de processo, a instância passa a ser executada e controlada, onde são alocadas as pessoas que realizam as suas atividades, utilizando recursos e produzindo produtos, por último a instância é concluída.

A Figura 5.1 apresenta um diagrama de atividades UML representando as atividades do negócio executar processos de software.



FIGURA 6.1.- Atividades do negócio executar processos.

6.2.- Modelo Conceitual

Nesta seção descreve-se as funcionalidades da máquina de processos de uma maneira mais detalhada, por meio da descrição de casos de uso.

6.2.1.- Objetivo do Sistema

Construir um sistema de software capaz de apoiar e/ou automatizar as atividades do negócio executar modelos de processos.

6.2.2.- Atores & Funcionalidades

Nesta seção serão especificados somente os atores externos (pessoas), que interagem com o sistema, e suas respectivas funções, outros tipos de atores como componentes de software serão descritos posteriormente.

Atores:

Gerente de Processos: pessoa responsável por configurar e monitorar instâncias de processos de software em um dado projeto.

Executor: pessoa diretamente envolvida na execução de atividades do processo, recebe e efetua tarefas em um dado projeto, onde produz artefato e utiliza recursos.

& suas funções

Gerente de Processos:

Planejar instâncias de processos

Monitorar a execução das instâncias

Executor:

Gerenciar tarefas

As próximas seções referem-se à visão use case do sistema.

6.2.3.- Visão Use Case: Contexto do Sistema

Nesta seção descreve-se o contexto da máquina de processos, módulo do serviço de coordenação de processos responsável pelo controle da execução de processos de software no ambiente e-WebProject. A descrição do contexto se dá pela definição de diagramas de caso de uso, devido a complexidade deste módulo, a definição de dá por meio de um diagrama de casos de uso de alto nível, representando a máquina de processos como um todo. Posteriormente estes casos de uso de mais alto nível serão expandidos para casos de uso de mais baixo nível que representam as funcionalidades reais da máquina de processos, a máquina de processos foi subdividida em três sub-módulos auxiliares sendo: dois responsáveis pela interação com os atores externos e um responsável pelo controle da execução dos processos.

A Figura 6.2 representa o diagrama de contexto da máquina de processos.

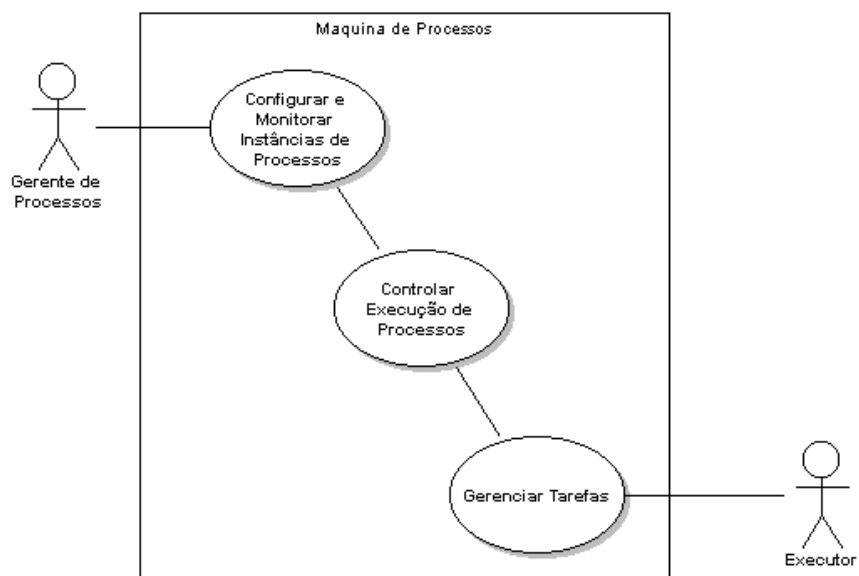


FIGURA 6.2.- Diagrama de Contexto: Máquina de Processos.

Como mencionado no parágrafo anterior a modelagem dos casos de uso, se dá em duas etapas, foi apresentado três casos de uso de alto nível representando as funções da máquina de processos e a interação com os atores externos, nesta segunda fase serão expandidos os casos de uso em um nível maior de detalhamento. Para cada caso de uso de alto nível foi definido um sub-módulo, responsável pela sua funcionalidade. Estes sub-módulos são os descritos abaixo:

- **Ferramenta de Configuração e Monitoração de Instâncias:** Módulo que auxilia o Gerente de Projeto a configurar e monitorar instâncias de processos de Software;
- **Espaço de Trabalho Eletrônico *Workspace*:** Módulo pelo qual o Desenvolvedor recebe uma lista de tarefas para seu gerenciamento e um conjunto de recursos para realização de seu trabalho.

- **Serviço de Agentes:** módulo central composto por agentes colaborativos, responsável pela criação e controle de execução de instâncias de processos de software.

A Figura 6.3 apresenta um diagrama de caso de uso de baixo nível, representando as funcionalidades da máquina de processos e a interação entre os submódulos.

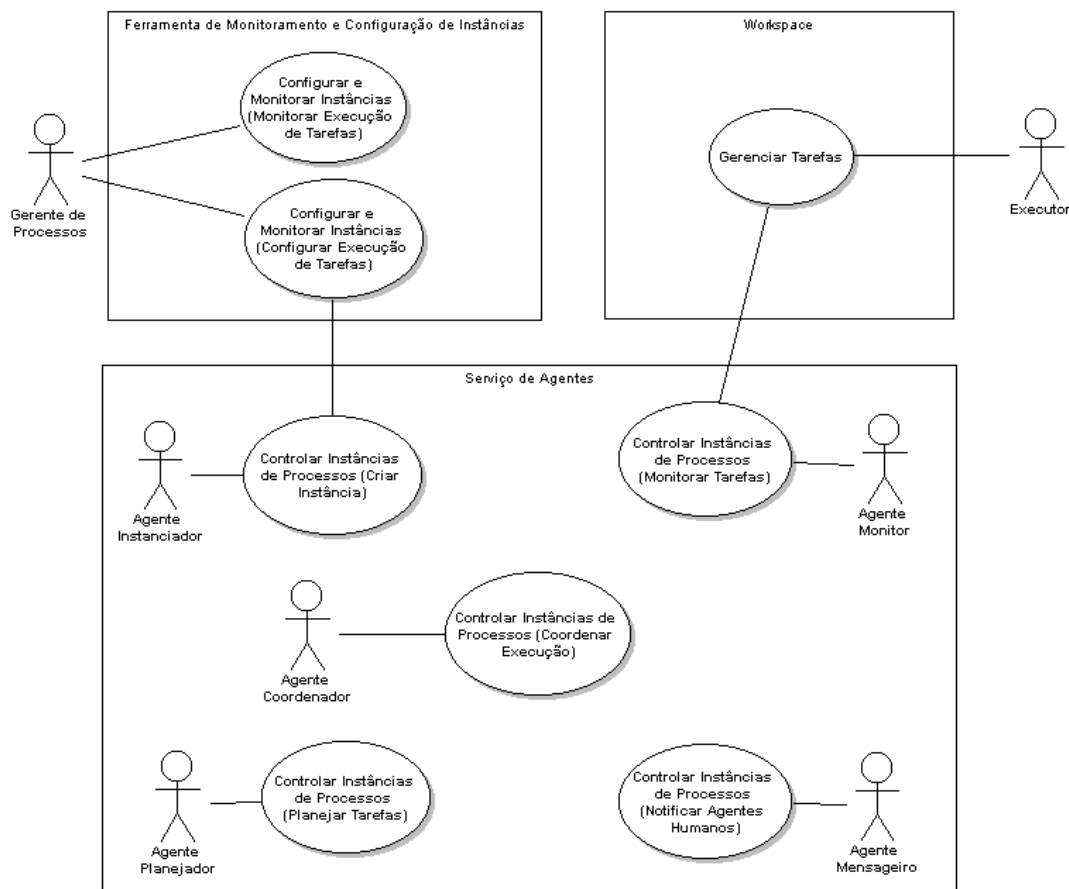


FIGURA 6.3.- Diagrama de Casos de Uso: sub-módulos da máquina de processos.

A descrição dos atores internos (agentes) e de cada caso de uso apresentado na Figura 6.3 vem a seguir, seguindo o mesmo formato apresentado no Capítulo 5. Primeiro a descrição dos agentes depois dos casos de uso.

6.2.4. Descrição dos Agentes

Os agentes definidos para o serviço de agentes são agentes autônomos colaborativos, agentes que fazem tarefas no intuito de realizar o caso de uso Controlar Instâncias de processos da máquina de processos. Segue a descrição de cada agente apresentado na Figura 6.3.

- **Agente Instanciador** - Agente responsável por criar novas instâncias de processos e definir as restrições de inicialização e conclusão de tarefas baseado nas regras definidas no modelo de processo.
- **Agente Coordenador** - Agente responsável por coordenar a execução das tarefas do processo.
- **Agente Planejador** - Agente responsável por planejar a execução de uma tarefa procurando a melhor opção dentre os Agentes Humanos (pessoas) disponíveis para exercer um determinado papel.
- **Agente Mensageiro** - Agente responsável por notificar as pessoas sobre a alocação de novas tarefas, ou alertar a gerência quando ocorrer algum problema.
- **Agente Monitor** - Agente responsável por monitorar as ações das pessoas que executam as tarefas, concluir as tarefas quando houver a solicitação e solicitar ao agente coordenador quando houver a conclusão de uma tarefa.

Uma descrição mais detalhada sobre como é feita a interação entre estes agentes e a realização de cada caso de uso será feita mais adiante.

6.2.5.- Descrição dos Casos de Uso

Segue a descrição do cada caso de uso Coordenar Instâncias de Processos (Coordenar Execução), uma descrição completa do conjunto de casos de uso, definidos para a máquina de processos encontra-se no apêndice D, para fins

de identificação são apresentadas as siglas para cada sub-módulo da máquina de processos:

MCM: Sigla para Ferramenta de Configuração e Monitoração de Instâncias;

MWS: Sigla para Espaço de Trabalho Eletrônico *Workspace*.

MAS: Sigla para Serviço de Agentes

6.2.5.1.- Caso de Uso: Controlar Instâncias de Processos (Coordenar execução)

Identificador do Caso de Uso	UC-MAS-02
Nome do Caso de Uso	Controlar Instâncias de Processos (Coordenar execução).
Atores	Agente Coordenador
Prioridade	Muito Alta
Pré Condição	Não há
Requisitos Especiais	Não há
<i>Primeiro Cenário – Coordenar execução Tarefas</i>	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o agente coordenador recebe uma mensagem para ativação de tarefas de processo; 2.- O agente coordenador avalia as pré condições satisfeitas das tarefas inativas; 3.- O agente coordenador envia uma mensagem para o agente mensageiro solicitando notificação de agentes humanos; 4.- O agente coordenador recebe uma mensagem do agente mensageiro e verifica se o estado da tarefa está "PRONTO" 5.- O agente coordenador atualiza os estados das tarefas para "ATIVO", e deposita na lista de tarefas do agente humano. 	
Fluxos Alternativos	
<p>No item 2, se não houverem agentes humanos planejados para a execução das tarefas, o agente coordenador envia uma mensagem para o agente planejador solicitando a melhor opção dentre os agentes humanos disponíveis para exercer os papéis da tarefa</p>	
Pós Condição:	
Não há	

6.3.- Modelo de Negócios

Nesta seção, inicia-se a análise e projeto dos requisitos funcionais descritos na seção 6.2. Será apresentado em primeiro lugar, a modelagem das classes de negócio que representam os aspectos estruturais da máquina de processos, posteriormente, serão apresentados aspectos comportamentais do sistema como modelagem de máquinas de estado, interação entre agentes e comportamento interno dos agentes.

6.3.1.- Modelagem Estrutural

São apresentadas nesta seção aspectos estruturais da máquina de processos, como modelagem de classes, atributos e serviços.

6.3.1.1.- Modelo de Classes Persistentes

A Figura 6.4 apresenta o diagrama de classes persistentes do repositório de instâncias de processos.

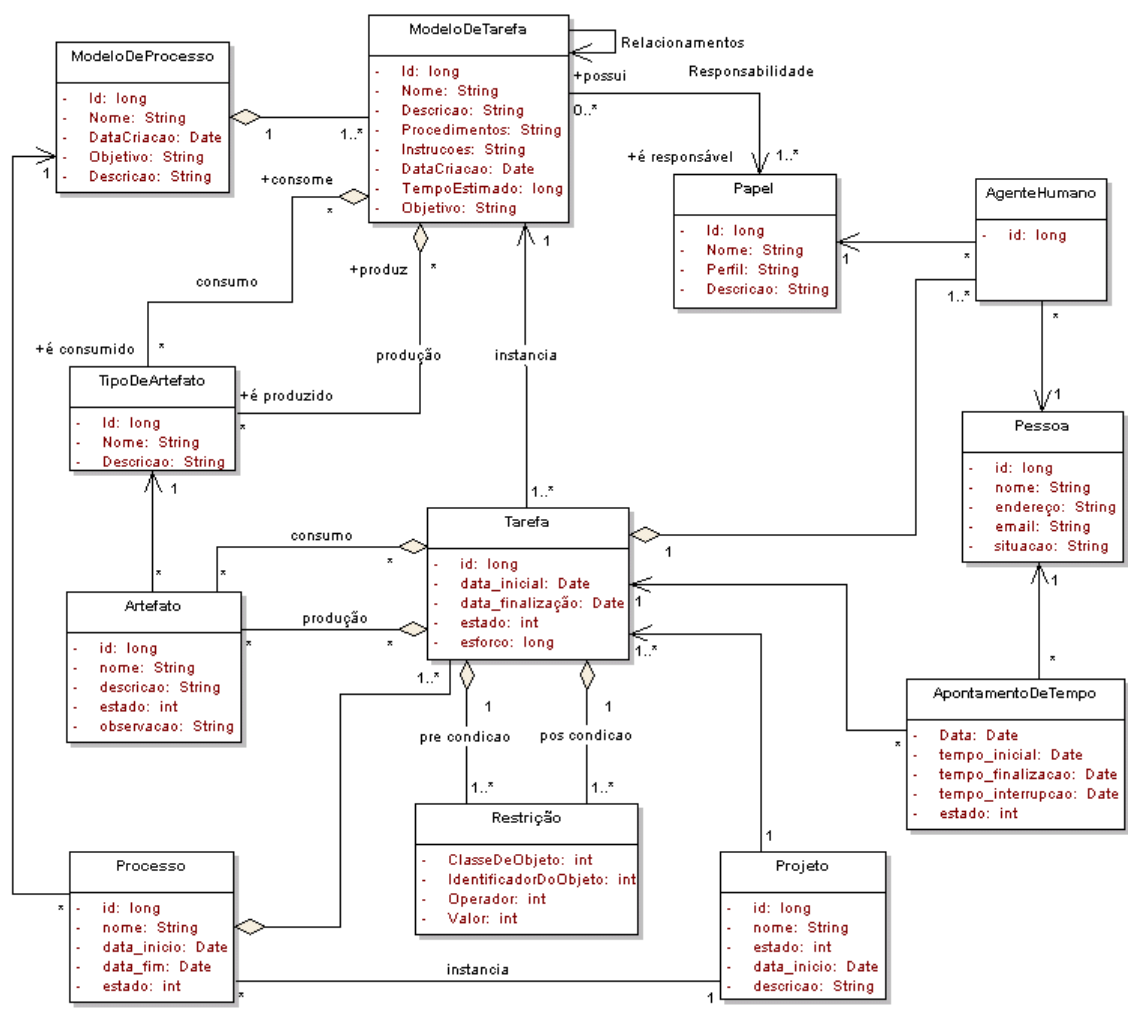


FIGURA 6.4.- Diagrama de classes persistentes: repositório de instâncias de processos.

As classes apresentadas no diagrama acima serão descritas na Tabela 6.1, as classes ModeloDeProcesso, ModeloDeTarefa, Papel e TipoDeArtefato já foram descritas na seção 5.3.1 do Capítulo 5.

TABELA 6.1.- Descrição das classes do repositório de instâncias de processos.

Classe	Descrição	Instâncias
Artefato	Um produto qualquer feito por uma pessoa envolvida em projeto em uma dada tarefa	Plano de teste para o caso de uso X, código fonte java da classe motor, revisão do plano de projeto x
Processo	Uma instância de um modelo de processo em um dado projeto	Engenharia de requisitos para o projeto X
Tarefa	Instância da classe TarefaModelo alocada em um dado projeto	Validar o protótipo do caso de uso X, revisar código fonte da classe Y
AgenteHumano	Uma pessoa que realize uma tarefa de um processo exercendo um determinado papel.	João da silva, desenvolve a classe Y, assumindo o papel de desenvolvedor Java
Pessoa	Uma pessoa qualquer que participe do projeto	-
ApontamentoDeTempo	Representa apontamentos de tempo para conclusão de uma tarefa em um dado projeto	-
Projeto	Um projeto de desenvolvimento de software qualquer	Precedência, sub-tarefa, feedback
Restrição	Representa uma pré condição para início de uma tarefa ou uma pós condição para a conclusão de uma tarefa	

As instâncias de tarefas em um dado projeto, são ativadas para execução pelo usuário de acordo com algumas restrições de entrada e saída que são descritas na próxima seção.

6.3.1.2.- Restrições para Ativação e Conclusão de Tarefas

As instâncias de tarefas em um dado projeto são ativadas ou concluídas, como pré ou pós condições de inicialização ou conclusão. Estas restrições são definidas de acordo com os relacionamentos definidos nos modelos de

processo. Cada relacionamento dá origem a uma restrição de inicialização ou conclusão de tarefa, definidas com pré e pós condições. Cada restrição é instância da classe Restrição, que possui os seguintes atributos descritos abaixo:

- **TipoDeObjeto:** representa qual classe de objeto a restrição é aplicada, como a própria classe Tarefa, a classe AgenteHumano, TipoDeArtefato e Artefato, por exemplo.
- **IdentificadorDoObjeto:** atributo que informa o identificador das instâncias das classes de objetos ao qual se aplica a restrição.
- **Operador:** serve para verificar se a restrição foi satisfeita ou não.
- **Valor:** valor para validação da restrição caso se aplique.

A Tabela 6.2 apresenta uma descrição de cada tipo de restrição imposta para inicialização e conclusão de tarefas.

TABELA 6.2.- Restrições aplicadas às tarefas.

Classe de objeto	Operador	Descrição	Aplicação
Tarefa	STATE_EQUALS	Indica que o estado de uma dada tarefa deve assumir algum valor específico. Ex. Tarefa "462" STATE_EQUALS "COMPLETADA"	Ativação
Tarefa	DATE_MUST_START_ON	Indica que a tarefa deve ser inicializada a partir de uma dada data. Ex.: Tarefa "537" DATE_MUST_START_ON "05/06/2004"	Ativação
Papel	AGENT_MUST_BE	Indica que o agente humano que vai	Ativação

(Continua)

Tabela 6.2 – Conclusão.

		<p>assumir este papel na execução da tarefa, deve ser uma pessoa específica. Ex. Papel “Engenheiro de Requisitos” AGENT_MUST_BE “João da Silva”</p>	
Papel	AGENT_ANY_AVAILABLE	<p>Indica que o agente humano que vai assumir este papel na execução da tarefa, pode ser qualquer pessoa disponível. Ex. Papel “desenvolvedor java” AGENT_ANY_AVAILABLE.</p> <p>OBS: quando ocorrer a validação deste tipo de restrição, o serviço deve localizar um agente disponível.</p>	ativação
TipoDeArtefato	ON_REPOSITORY	<p>Indica que um tipo de artefato deve estar armazenado no repositório. Ex.: Tipo de artefato “Código fonte Java” ON_REPOSITORY “452”</p>	ativação/ finalização
TipoDeRecurso	RESOURCE_AVAILABLE	<p>Indica que um recurso específico deve estar disponível para utilização. Ex.: TipoDeRecurso “scanner de mesa” RESOURCE_AVAILABLE</p>	ativação

6.3.1.3.- Modelo de Classes do Serviço de Agentes

A Figura 6.5 apresenta o diagrama de classes definidas para o serviço de agentes, incluindo as classes dos agentes e comportamentos.

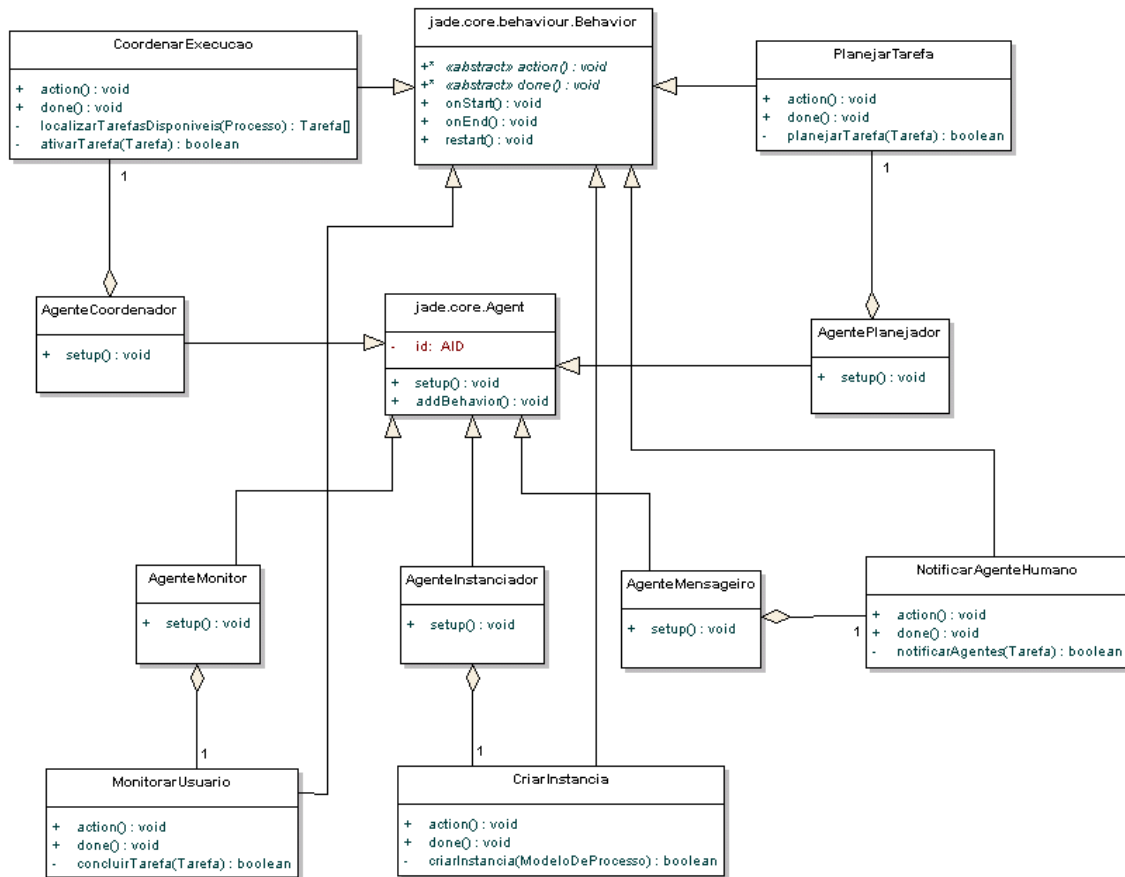


FIGURA 6.5.- Diagrama de classes: serviço de Agentes.

A descrição das classes do Serviço de Agentes podem ser vistas na Tabela 6.3

TABELA 6.3.- Descrição das classes do serviço de agentes.

Classe	Descrição
jade.core.Agent	Superclasse que define as características de um Agente na plataforma jade.
Jade.core..behaviour.Behavior	Superclasse que representa um comportamento de agente na plataforma jade.
AgenteInstanciador	Classe que representa o agente instanciador.
AgenteCoordenador	Classe que representa o agente coordenador.
AgentePlanejador	Classe que representa o agente planejador.
AgenteMonitor	Classe que representa o agente monitor.
AgenteMensageiro	Classe que representa o agente mensageiro.
CriarInstancia	Classe que define o comportamento do agente instanciador
CoordenarExecução	Classe que define o comportamento do agente coordenador
PlanejarTarefa	Classe que define o comportamento do agente planejador
MonitorarTarefa	Classe que define o comportamento do agente mensageiro
Notificar Agente Humano	Classe que define o comportamento do agente monitor

6.3.2.- Modelagem Comportamental

Nesta seção são apresentados aspectos comportamentais da máquina de processos como estados de objetos, interação e comunicação entre agentes e modelagem de comportamento interno.

6.3.2.1.- Os Estados de um Processo

Uma instância de processo assume alguns estados, que determinam em que fase o processo se encontra, isto pode ser indicado através do valor do atributo estado da classe Processo. A modelagem dos estados de um processo pode ser vista na Figura 6.6.

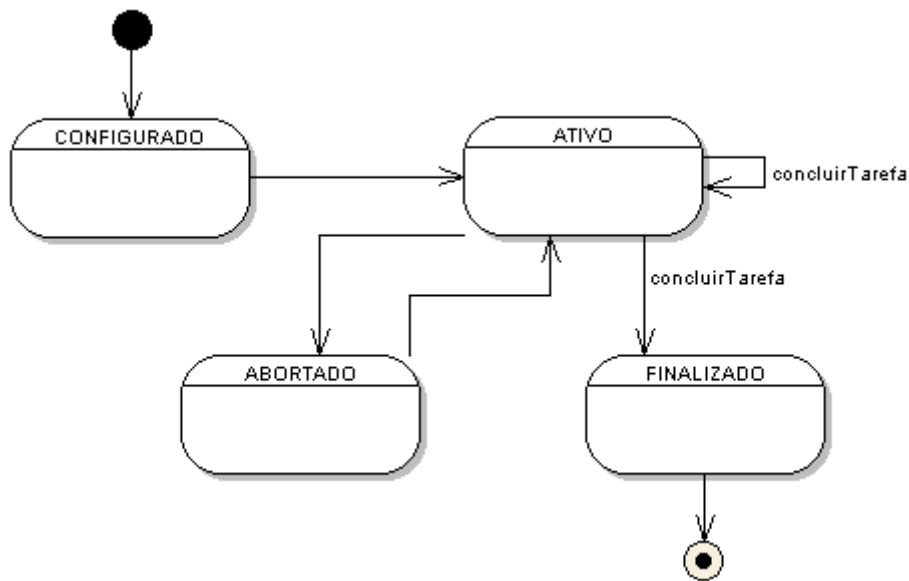


FIGURA 6.6.- Estados de uma instância de processo.

A descrição de cada estado de processo pode ser vista na Tabela 6.4.

TABELA 6.4.- Descrição dos estados de um processo.

Estado	Descrição
CONFIGURADO	Processo configurado e pronto para ser ativado
ATIVO	O processo está em execução
FINALIZADO	A instância do processo foi concluída.
ABORTADO	Ocorreu algum problema na execução do processo.

6.3.2.2.- Os Estados de uma Tarefa

A tarefa é o principal componente de um processo, através dela orientamos a execução de um processo. Como um processo é composto por diversas tarefas, e estas tarefas são executadas ao longo do tempo, considera-se certas condições de disparo e finalização. A Tarefa pode assumir alguns estados que representam em qual fase ela se encontra. Os valores do atributo estado indicam a situação das tarefas que podem ser vistos na Figura 6.7.

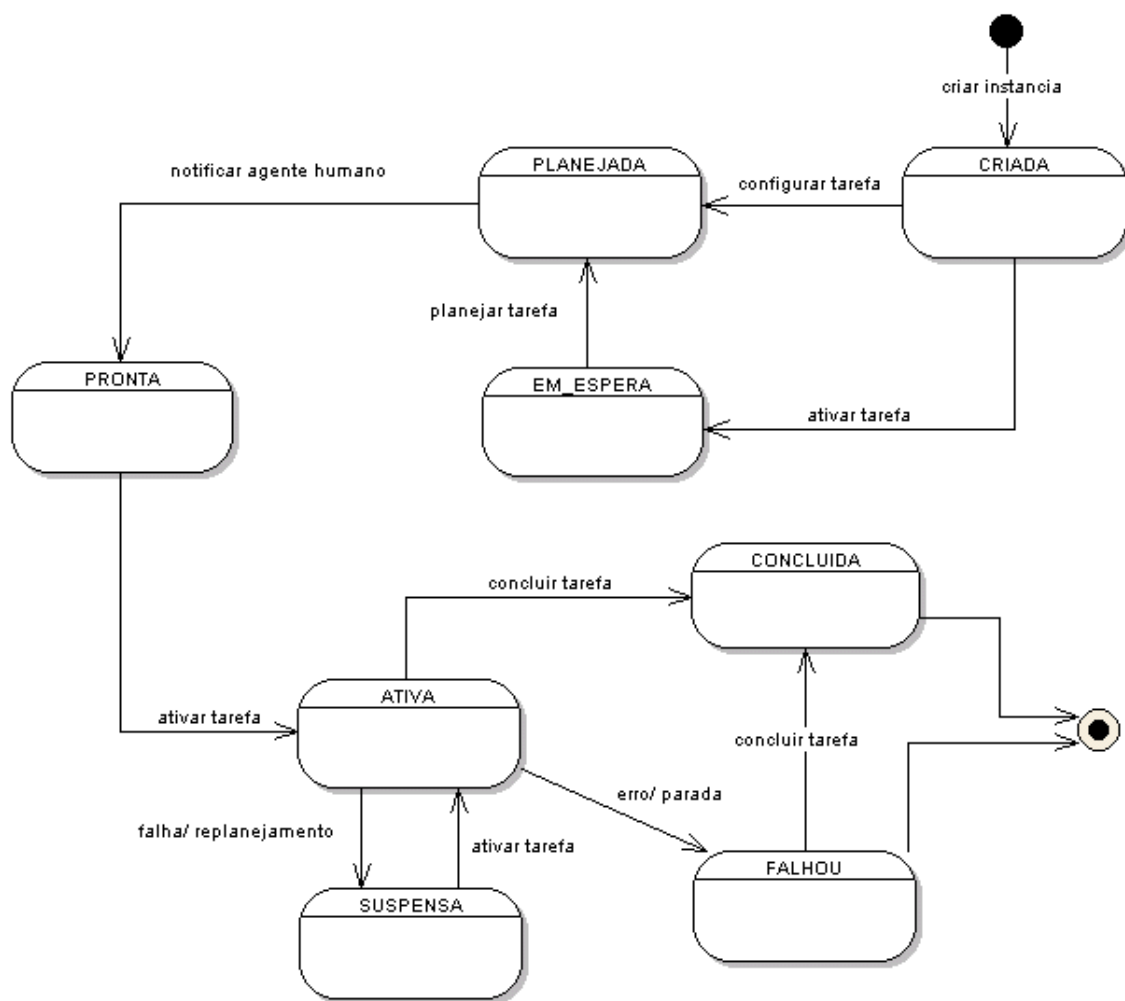


FIGURA 6.7.- Estados de uma tarefa.

A descrição de cada estado de uma tarefa pode ser vista na Tabela 6.5.

TABELA 6.5.- Descrição dos estados de uma tarefa.

Estado	Descrição
CRIADA	A tarefa foi criada, mas não existem agentes humanos definidos para sua execução
EM_ESPERA	A Instancia está esperando que as restrições de inicialização sejam satisfeitas e não existem agentes humanos configurados para a execução de tarefas
PLANEJADA	Indica que um ou mais agentes humanos foram configurados para executar a tarefa, mas as restrições de de inicialização ainda não foram satisfeitas.
PRONTA	Indica que os agentes humanos foram notificados e a tarefa está pronta para ser ativada.
ATIVA	A tarefa está ativa, sendo executada por algumas pessoas.
SUSPENSA	A tarefa entra em estado de suspensão, quando ocorre uma falha ou necessita re-planejamento.
CONCLUÍDA	Indica que a tarefa foi concluída e as restrições de saída foram satisfeitas.
FALHOU	Indica que houve alguma falha quando na conclusão das tarefas, como restrições de saída não satisfeitas.

6.3.2.3.- O Protocolo de Execução de Processos

Dá-se início a execução de uma instância de modelo de processo, a partir de uma solicitação de um gerente de projeto, que pode configurar agentes humanos para executar as tarefas ou não.

A partir deste momento a solicitação é encaminhada ao agente instanciador que verifica as configurações associadas as regras definidas no respectivo modelo de processo encontrado no repositório, o agente instanciador define então as tarefas e restrições de entrada e saída, ativa a instancia do processo e notifica ao agente coordenador a existência da nova instância.

O agente coordenador, por sua vez, localiza as tarefas disponíveis, valida as restrições de entrada das tarefas, verifica necessidade de alocação de agentes humanos, caso haja a necessidade, solicita planejamento da tarefa ao agente planejador.

O agente planejador procura as opções de pessoas associadas aos papéis de execução das tarefas e aloca os agentes humanos à tarefa, por fim o agente planejador informa ao agente coordenador o planejamento da tarefa o agente coordenador solicita notificação aos agentes humanos que executarão a tarefa ao agente mensageiro, e ativa as tarefas.

A tarefa é então colocada na lista de tarefas dos desenvolvedores, que avaliam as instruções, apontam tempos de execução e produzem artefatos. Quando finalizadas as tarefas, os desenvolvedores marcam a tarefa como “COMPLETADA”, este evento é notificado ao agente monitor.

O agente monitor valida as restrições de saída da tarefa, após validar as restrições o agente monitor atualiza o estado da tarefa para completado, e verifica se existem tarefas que não foram completadas no processo, caso hajam notifica ao agente coordenador que refaz o seu trabalho, caso não, atualiza o estado do processo para “COMPLETADO”.

A Figura 6.8, apresenta um diagrama de atividades que mostra o protocolo de execução de processos, e a interação entre os agentes do sistema.

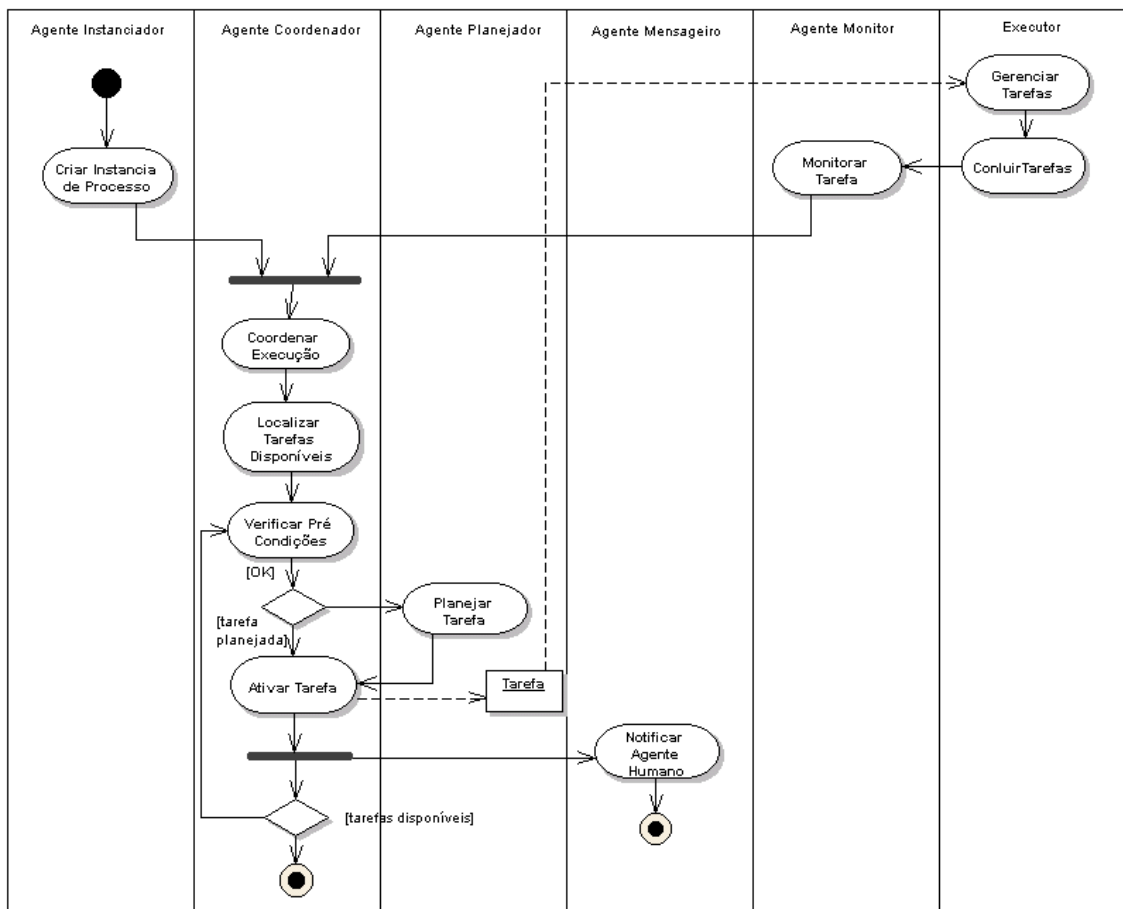


FIGURA 6.8.- Diagrama de atividades que apresenta o protocolo de execução de processos entre os agentes.

6.3.2.4.- Modelagem da Comunicação entre Agentes

A comunicação entre os agentes se dá através de mensagens, que seguem uma linguagem de comunicação de agentes especificada pela FIPA, na plataforma JADE, existe uma classe de objeto ACLMessage, que define uma mensagem que pode ser encaminhada a um agente da plataforma. A ação do agente pode ser disparada através da chegada de uma mensagem. A linguagem de comunicação de agentes, também disponível na plataforma, permite facilidades para validação e tratamento de mensagens.

A figura 6.9, apresenta a colaboração entre os agentes que compõem a máquina de processos, entenda-se como agentes os atores internos e externos do sistema.

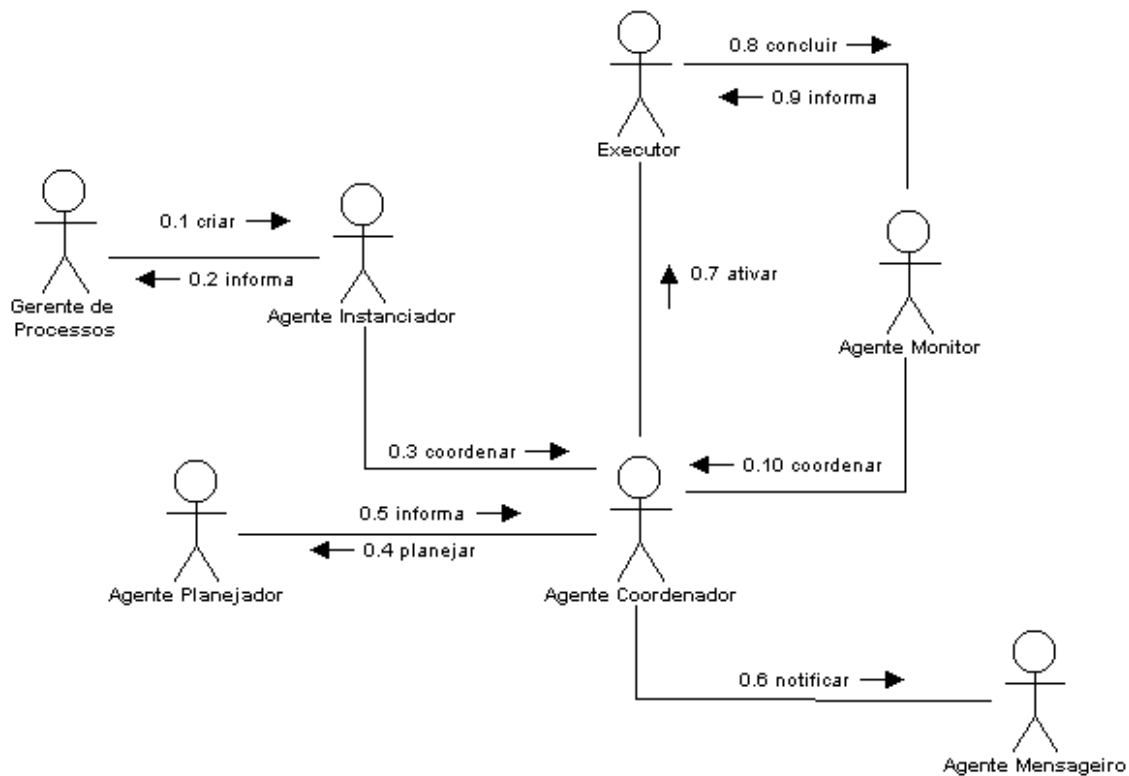


FIGURA 6.9.- Diagrama de colaboração: interação entre os agentes.

A Figura 6.10 apresenta um diagrama de seqüência que mostra a ordem temporal das trocas de mensagens feitas pelos agentes. A descrição das mensagens apresentadas nas figuras 6.9 e 6.10 será dada a seguir:

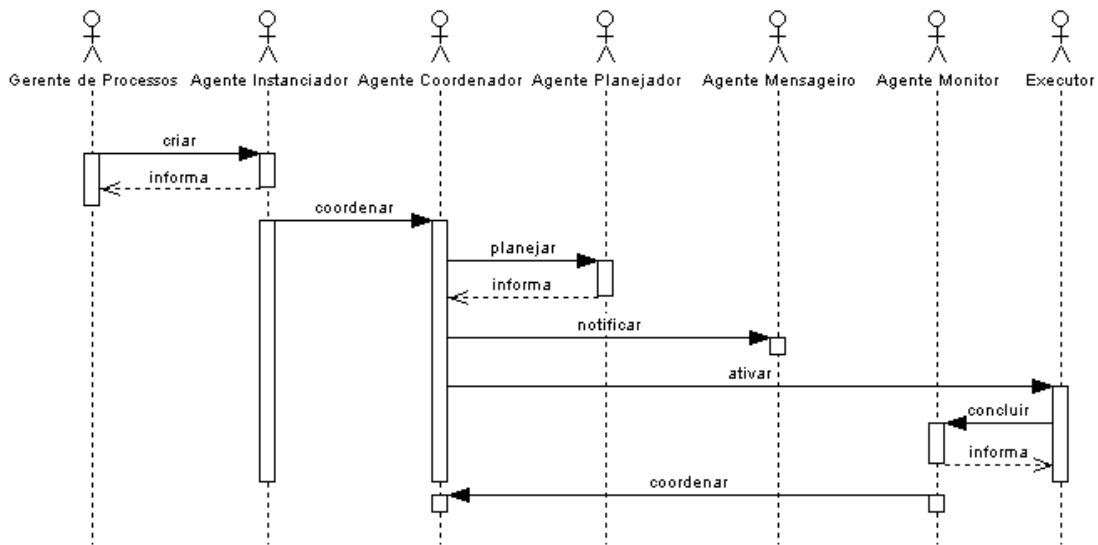


FIGURA 6.10.- Diagrama de seqüência: troca de mensagens entre os agentes.

Descrição da seqüência das mensagens:

- 1) O gerente de projetos solicita ao agente instanciador, a criação de uma nova instância de processo;
- 2) O agente instanciador cria a instância e informa ao gerente de projetos;
- 3) O agente instanciador solicita ao agente coordenador, que ative as tarefas do processo;
- 4) O agente coordenador verifica se existe a necessidade de alocação de agentes humanos à tarefa e solicita ao agente planejador que planeje a execução tarefa;
- 5) O agente planejador planeja a tarefa e informa o agente coordenador;
- 6) O agente coordenador solicita ao agente mensageiro que notifique as pessoas que executarão as tarefas;
- 7) O agente coordenador ativa a tarefa e coloca na lista de tarefas dos desenvolvedores;

- 8) O desenvolvedor executa a tarefa e solicita conclusão ao agente monitor.
- 9) O agente monitor, verifica as restrições de saída, conclui a tarefa, informa o resultado ao desenvolvedor e verifica se ainda existem tarefas a ser concluídas no processo, caso não, conclui o processo.
- 10) O agente monitor solicita ao agente coordenador que coordene os próximos passos da execução do processo.

6.3.2.5.- Modelagem do Processamento Interno dos Agentes

Nesta seção são apresentados aspectos internos do comportamento dos agentes. O comportamento interno dos agentes são modelados, utilizando-se diagramas de atividades UML, esta abordagem pode ser vista em (Odell e outros 2000), são utilizados dois novos elementos que foram incluídos na especificação UML para modelar eventos em diagramas de atividades. Os elementos utilizados são respectivamente: *Send Event*, um retângulo com um “V” deitado do lado direito que é utilizado para modelar geração de estímulos externos, o outro elemento é o *Receive Event*, um retângulo com um “V” deitado do lado esquerdo, que é utilizado para modelar a recepção de estímulos externos ao sistema. Foram utilizados estes elementos devido as características de comunicação dos agentes.

Segue a modelagem do comportamento interno dos agentes, mostrando as atividades internas de cada um nas próximas sub seções a seguir.

6.3.2.5.1.- O Comportamento Interno do Agente Instanciador

O agente instanciador inicia seu trabalho a partir da solicitação de criação de uma nova instância de processo, por parte do gerente de projeto. Isto se dá através da chegada uma mensagem à caixa de mensagens do agente. O agente então verifica o modelo de processo, cria as tarefas, define as restrições de entrada e saída, configura a execução das tarefas, ativa a

instância do processo e solicita ao agente coordenador que coordene a sua execução.

O comportamento interno do agente instanciador é mostrado na Figura 6.11.

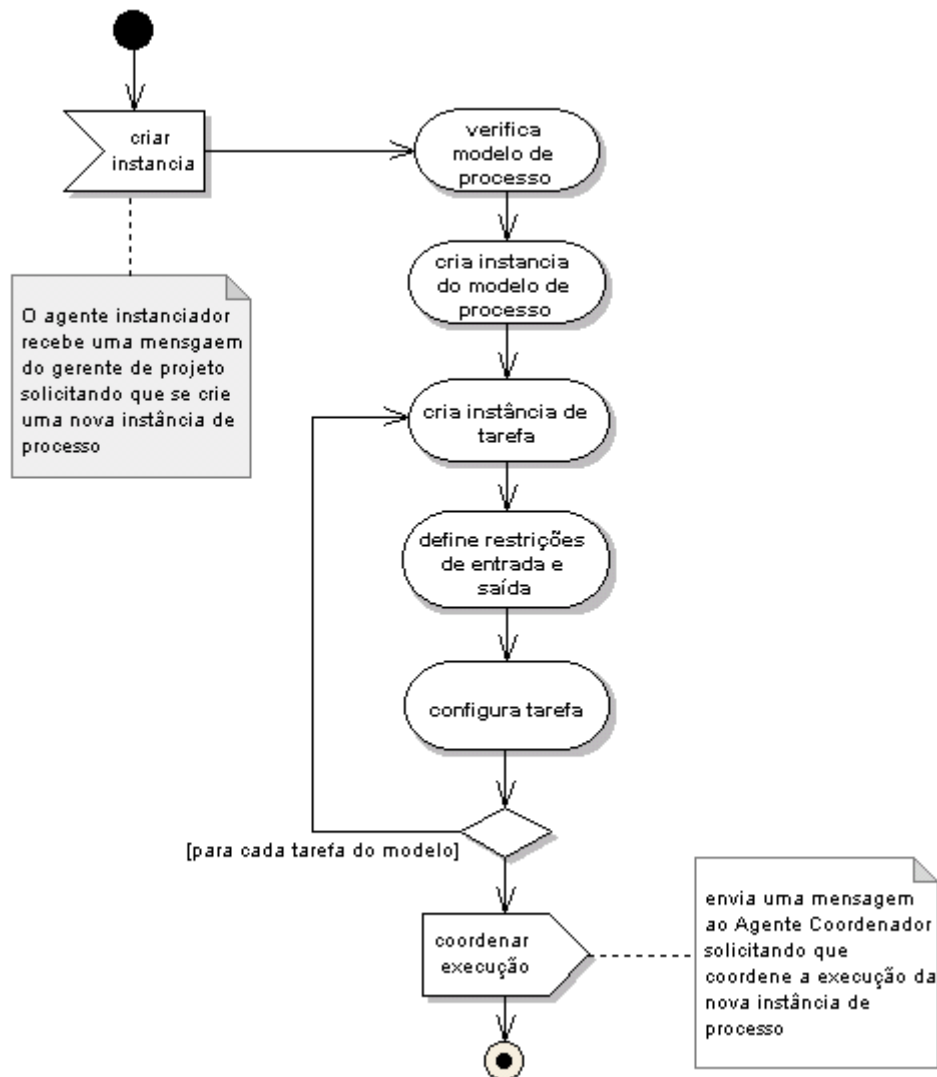


FIGURA 6.11.- Diagrama de atividades mostrando o comportamento interno do agente instanciador.

6.3.2.5.2.- O Comportamento Interno do Agente Coordenador

O agente coordenador inicia seu trabalho a partir de uma mensagem recebida do agente instanciador ou do agente monitor, o agente coordenador verifica, as tarefas que podem ser ativadas, avaliando as restrições de entrada, verifica necessidade de alocação de pessoas, caso haja, solicita planejamento ao agente planejador através de uma mensagem, bloqueia a execução até o agente planejador responder a mensagem, o agente coordenador ativa a tarefa e solicita ao agente mensageiro que notifique as pessoas da ativação, o agente coordenador finaliza o seu trabalho.

O comportamento interno do agente coordenador pode ser visto na Figura 6.12.

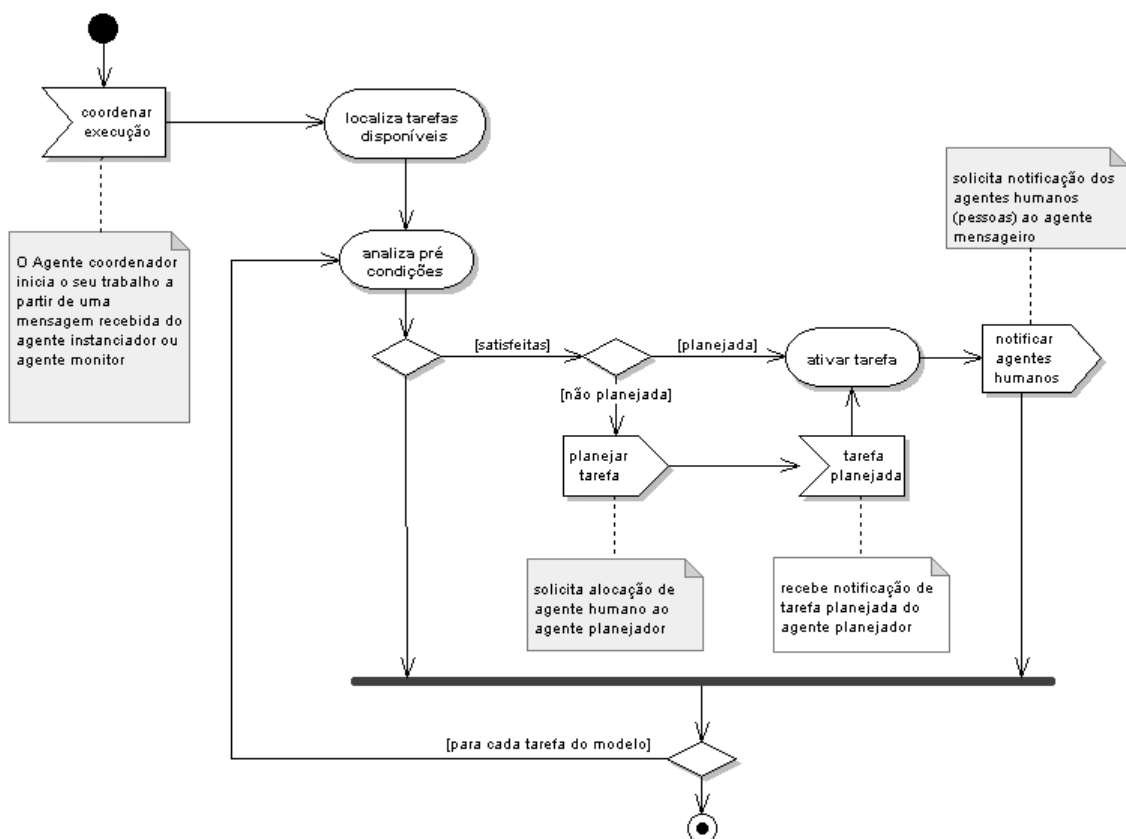


FIGURA 6.12.- Diagrama de atividades mostrando o comportamento interno do agente coordenador.

6.3.2.5.3.- O Comportamento Interno do Agente Planejador

O agente planejador inicia o seu trabalho a partir de uma mensagem recebida do agente coordenador, o agente planejador, localiza a tarefa modelo da instância da tarefa, localiza os papéis responsáveis pela execução da tarefa modelo, para cada papel, verifica quais pessoas possuem perfil para assumir aquele papel, escolhe a pessoa com menos tarefas alocadas, associa a pessoa ao papel e aloca o agente humano à tarefa, finalizado o planejamento da tarefa, atualiza o estado da tarefa para “PLANEJADA”, e responde a mensagem ao agente coordenador.

O comportamento interno do agente planejador pode ser visto na Figura 6.13.

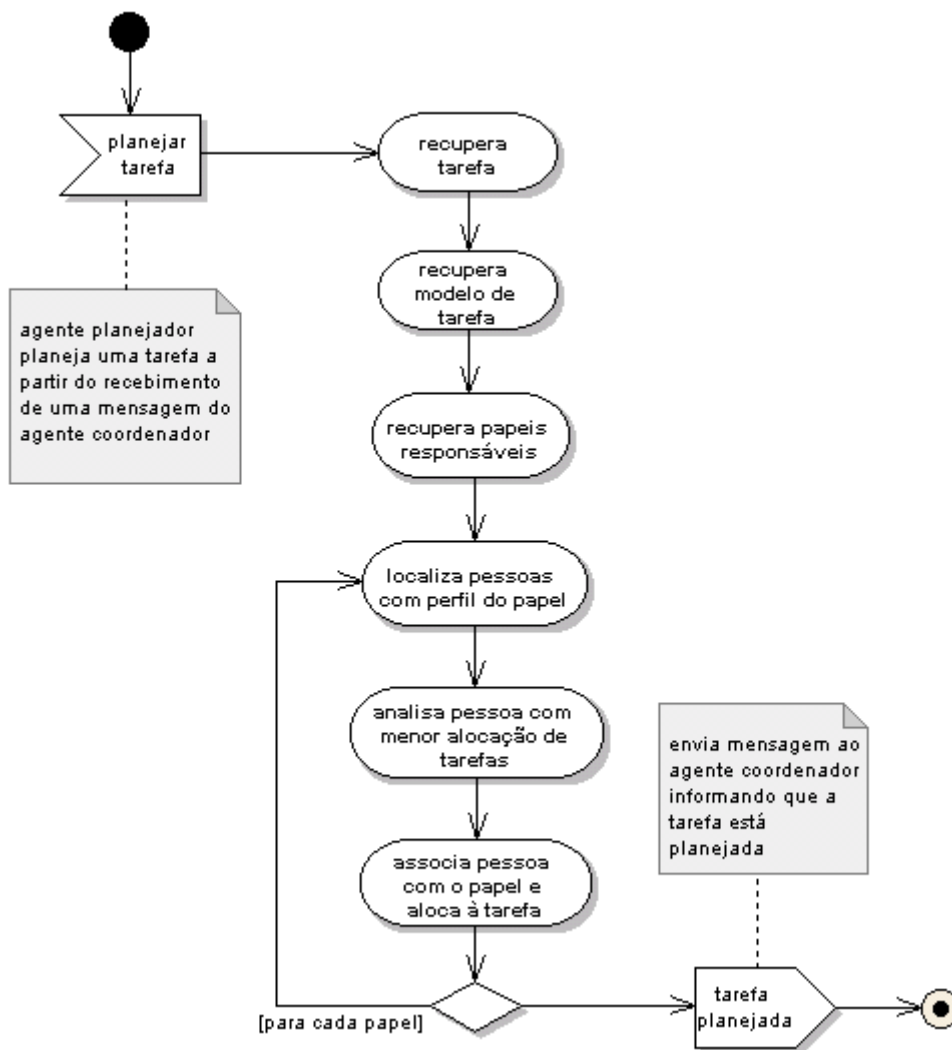


FIGURA 6.13.- Diagrama de atividades mostrando o comportamento interno do agente planejador.

6.3.2.5.4.- O Comportamento Interno do Agente Mensageiro

O agente mensageiro inicia seu trabalho a partir de uma mensagem do agente coordenador solicitando notificação de agentes humanos, o agente mensageiro, recupera a tarefa da mensagem, monta uma lista de endereços eletrônicos das pessoas alocadas à tarefa, monta uma mensagem de texto informando sobre a ativação da tarefa, cria uma mensagem SMTP, associa a lista de endereços e a mensagem de texto, conecta a um servidor SMTP e envia a mensagem.

O comportamento interno do agente mensageiro pode ser visto na Figura 6.14.

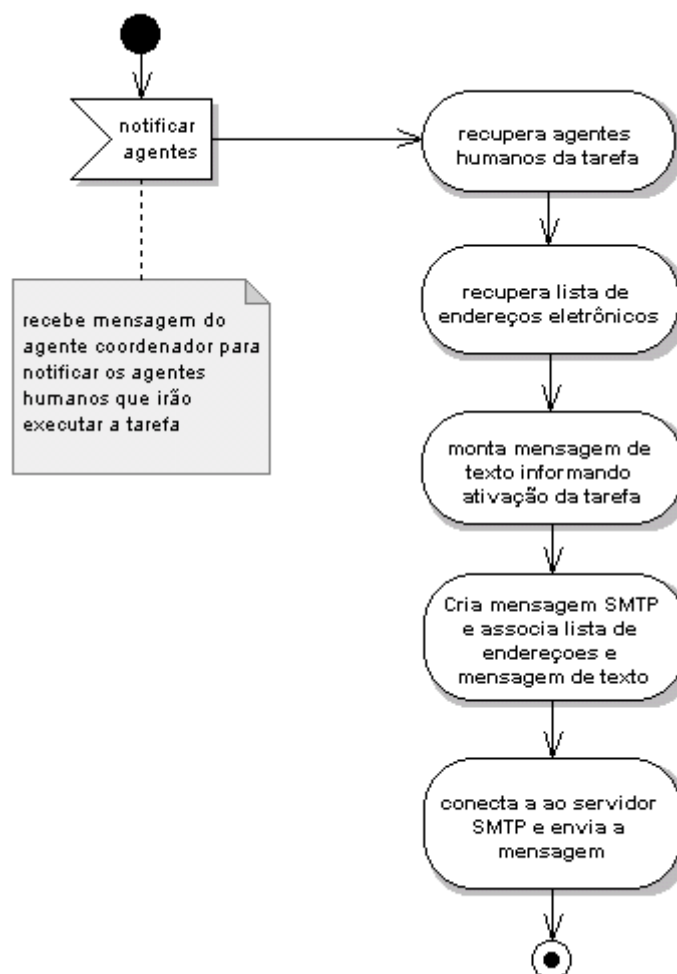


FIGURA 6.14.- Diagrama de atividades mostrando o comportamento interno do agente mensageiro.

6.3.2.5.5.- O Comportamento Interno do Agente Monitor

O agente monitor inicia seu trabalho a partir da chegada de uma mensagem do desenvolvedor da tarefa, o agente monitor avalia as restrições de finalização da tarefa, atualiza o estado da tarefa para “COMPLETADA”, e verifica se existem tarefas a serem ativadas no processo, caso haja tarefas, solicita ao agente coordenador que coordene os próximos passos da execução do processo, caso não, atualiza o estado do processo para “COMPLETADO”.

O comportamento interno do agente monitor pode ser visto na Figura 6.15.

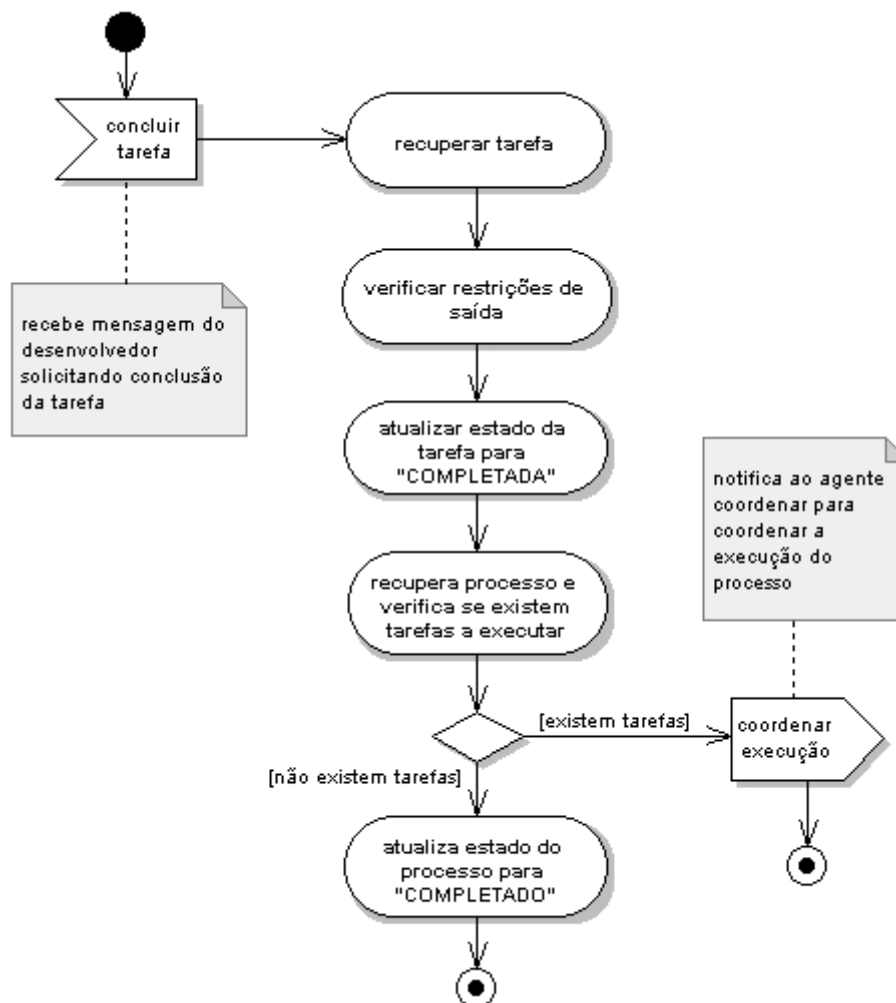


FIGURA 6.15.- Diagrama de atividades mostrando o comportamento interno do agente monitor.

No próximo capítulo são apresentados os aspectos de modelagem do repositório de artefatos, módulo do serviço de coordenação de processos responsável pelo controle dos produtos produzidos nas instâncias de processos.

CAPÍTULO 7

O REPOSITÓRIO DE ARTEFATOS

Neste capítulo, descreve-se as fases de análise de requisitos e modelagem do repositório de artefatos, módulo do serviço de coordenação de processos responsável pelo controle dos artefatos (produtos) produzidos durante a execução de instâncias de processos no e-WebProject.

Em primeiro lugar, define-se o escopo do problema a ser resolvido, posteriormente são descritos os casos de uso, e a modelagem das classes de negócio deste módulo, o repositório de artefatos não foi implementado no protótipo inicial do serviço de coordenação de processos, apesar de sua importância, por esta razão não serão apresentados níveis mais baixos de modelagem como modelagem do comportamento do sistema limitando-se neste capítulo a descrição das funcionalidades do sistema e a modelagem das classes persistentes.

7.1.- Modelo Descritivo

Descreve-se nesta seção, o escopo do problema a ser resolvido com o desenvolvimento do repositório de artefatos.

7.1.1.- Domínio do Negócio

Controle de artefatos produzidos em instâncias de modelos de processos de software.

7.1.2.- Descrição do Negócio

Durante a execução de um processo de software diversos artefatos são produzidos por diversas pessoas envolvidas em um projeto de software, estes artefatos por sua vez, possuem características bem definidas e devem ser produtos de atividades de um processo de desenvolvimento de software bem definido e formalizado em uma organização.

Estes artefatos podem ser qualquer coisa produzida por alguém envolvido no processo e que tenha utilidade durante o desenvolvimento de um projeto de software como por exemplo: planos de desenvolvimento, documentos de requisitos, planos de teste, código fonte, componente reutilizável, dentre outros.

Software é mais do que código fonte ou código executável. Um documento de requisitos bem desenvolvido pode ser base para uma negociação com o cliente sobre quais funcionalidades devem ter prioridade no desenvolvimento, um relatório de testes, por exemplo, que utilizou dados de diversos casos de testes, produzidos durante a execução do processo de desenvolvimento, pode validar uma arquitetura de desenvolvimento de software, indicando que ela atende aos requisitos não funcionais definidos no escopo do projeto ou servir de argumento para a troca da mesma. Em outras palavras, os artefatos possuem um papel fundamental no desenvolvimento de um sistema de software já que eles são o próprio sistema de software.

Os artefatos também fazem parte do fluxo de trabalho de diversos processos de software, servindo como entrada para uma atividade de um processo ou sendo produto da mesma.

Devido a toda esta importância, alguns cuidados devem ser tomados, um artefato pode possuir dependência com outros artefatos, o que pode ter um grande impacto caso haja necessidade de modificação, a evolução de um artefato é algo natural no contexto do desenvolvimento de software, portanto, um controle de versões adequado deve ser aplicado, a sobreposição de versões também se torna algo que deve ser evitado, para tanto, funcionalidades como check-in e check-out devem ser disponibilizadas.

Um repositório de artefatos deve então, manter seus artefatos, versões e manter um efetivo controle de versões além do controle de configurações.

A Figura 7.1 representa as atividades do negócio controlar artefatos.

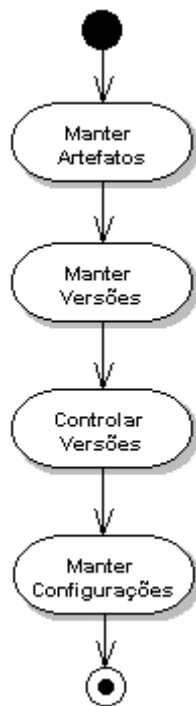


FIGURA 7.1.- Atividades do negócio controlar artefatos.

7.2.- Modelo Conceitual

Nesta seção descreve-se as funcionalidades do repositório de artefatos de uma maneira mais detalhada, por meio da descrição de casos de uso.

7.2.1.- Objetivo do Sistema

Construir um sistema de software capaz de apoiar e/ou automatizar as atividades do negócio controlar artefatos.

7.2.2.- Atores & Funcionalidades

Nesta seção serão especificados os atores externos (pessoas), que interagem com o sistema e suas respectivas funções.

Atores:

Gerente de Configuração: pessoa responsável por gerenciar o repositório sendo assim, responsável por manter configurações de releases de produtos,

congelar versões de artefatos, manter artefatos e versões, além de definir e manter novos tipos de artefatos.

Executor: pessoa diretamente envolvida na execução de atividades do processo, recebe e efetua tarefas em um dado projeto, onde produz artefatos e utiliza recursos.

& suas funções

Gerente de Configuração:

Mantém Tipos de Artefatos

Mantém Artefatos

Mantém Versões

Mantém Configurações

Congela versões de artefatos

Executor

Cria Novo Artefato

Cria Nova Versão

Check-In de Versão

Check-Out de Versão

Manter Histórico de Modificação

As próximas seções referem-se à visão use case do sistema.

7.2.3.- Visão Use Case: Contexto do Sistema

Nesta seção descreve-se o contexto do repositório de artefatos, representando as funcionalidades através de um diagrama de casos de uso, mostrado na Figura 7.2.

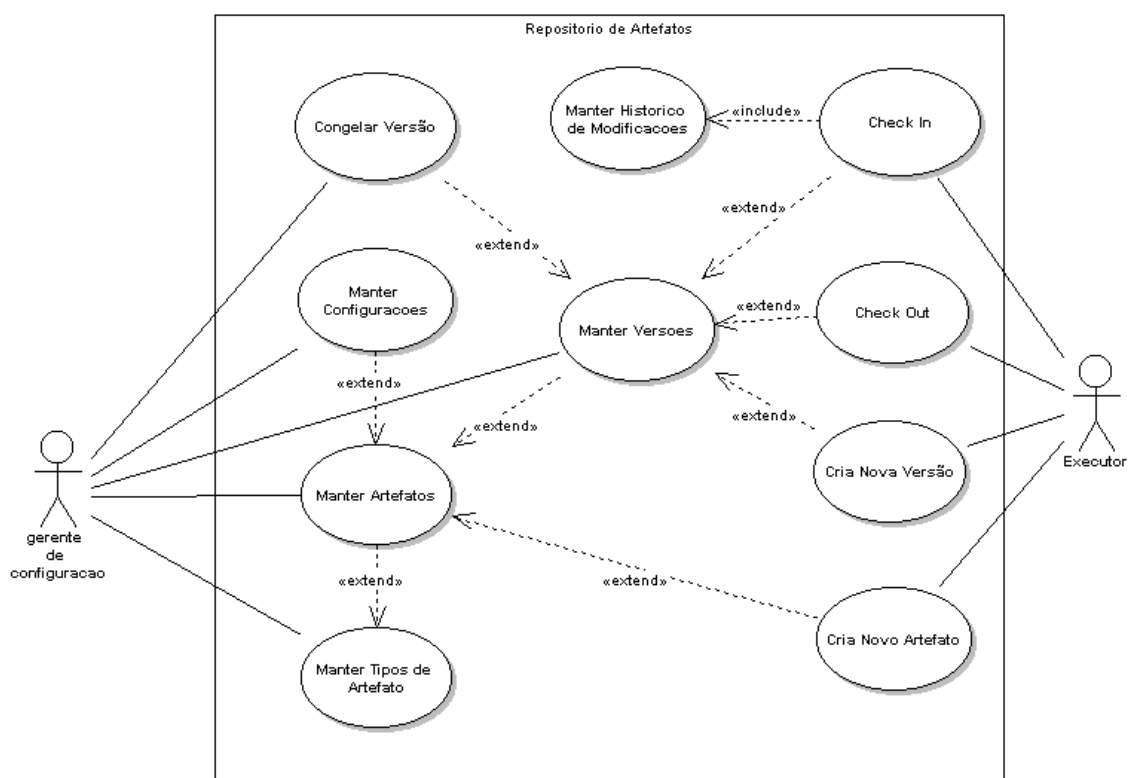


FIGURA 7.2.- Diagrama de Casos de Uso: Repositório de Artefatos.

7.2.4.- Descrição dos Casos de Uso

Segue a descrição do caso de uso Manter Artefatos. Uma descrição completa do conjunto de casos de uso, encontra-se no apêndice E, para fins de identificação é apresentada a sigla para o sub-módulo do serviço de coordenação de processos: o repositório de artefatos:

ARP: Sigla que identifica as funcionalidades do Repositório de Artefatos;

7.2.4.1.- Caso de Uso: Manter Artefatos

Identificador do Caso de Uso	UC-ARP-02
Nome do Caso de Uso	Manter Artefatos
Atores	Gerente de Configuração
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Gerente de Configurações deve ter sido validado pelo sistema O caso de uso UC-ARP-01 deve ter sido executado previamente
Requisitos Especiais	Não há
Primeiro Cenário – Criar Novo Artefato	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface de navegação de artefatos; 2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo; 3.- O gerente de configurações seleciona um tipo de artefato; 4.- O sistema apresenta uma tela do lado direito informando os dados do tipo de artefato e oferecendo uma opção de criar um artefato baseado naquele tipo; 5.- O Gerente de Configuração seleciona a opção apresentada; 6.- O sistema apresenta um formulário com os dados referentes ao artefato, como nome, data de criação, descrição e observação; 7.- O gerente de configurações entra com os dados; 8.- O gerente de configurações submete as informações; 9.- O sistema verifica as informações e salva na base de dados. 	
Fluxos Alternativos	
No passo 9, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário	
Pós Condição:	
O sistema apresentará mensagem de sucesso da operação.	
Segundo Cenário – Editar Artefato	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface de navegação de artefatos; 2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo; 3.- O Gerente de Configuração expande a árvore de artefatos e seleciona o artefato que deseja editar as informações; 4.- O sistema apresenta uma tela com as informações do artefato, apresentando algumas opções, dentre elas a de editar os dados do artefato; 5.- O gerente de configurações seleciona a opção “Edit”; 6.- O sistema apresenta um formulário com os dados atuais do artefato; 7.- O gerente de configurações atualiza as informações do artefato; 8.- O gerente de configurações submete as informações; 9.- O sistema verifica as informações e salva na base de dados. 	
Fluxos Alternativos	
No passo 9, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao	

usuário
Pós Condição: O sistema apresentará mensagem de sucesso da operação.
Terceiro Cenário – Excluir Artefato
Fluxo de Eventos:
Fluxo Principal 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface de navegação de artefatos; 2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo; 3.- O Gerente de Configuração seleciona expande a árvore de artefatos e seleciona o artefato que deseja editar as informações; 4.- O sistema apresenta uma tela com as informações do artefato, apresentando algumas opções, dentre elas a de excluir o artefato; 5.- O gerente de configurações seleciona a opção "Delete"; 6.- O sistema apresenta um formulário de confirmação de exclusão; 7.- O gerente de configurações confirma a exclusão e submete as informações; 8.- O sistema verifica as informações e exclui da base de dados.
Fluxos Alternativos No passo 8, se ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário
Pós Condição: O sistema apresentará mensagem de sucesso da operação.

7.3.- Modelo de Negócios

Nesta seção, inicia-se a análise e projeto das funcionalidades descritas na seção 7.2. Será apresentado a modelagem das classes de negócio que representam os aspectos estruturais do repositório de artefatos.

7.3.1.- Modelagem Estrutural

São apresentadas nesta seção aspectos estruturais do repositório de artefatos, como modelagem de classes, atributos e serviços.

7.3.1.1.- Modelo de Classes Persistentes

A Figura 7.3 apresenta o diagrama de classes persistentes do repositório de artefatos.

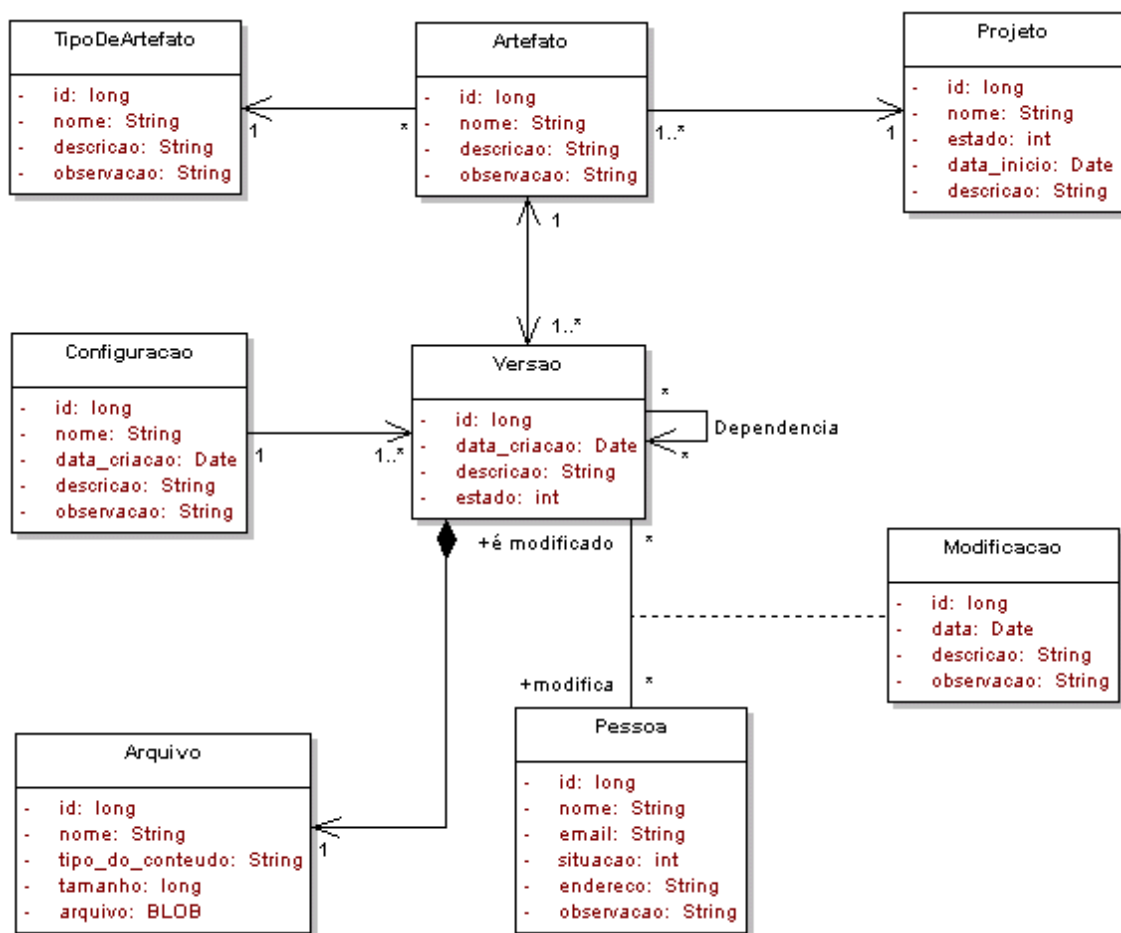


FIGURA 7.3.- Diagrama de classes: repositório de artefatos.

A Tabela 7.1 apresenta uma descrição de cada classe do repositório:

TABELA 7.1.- Descrição das classes do repositório de artefatos.

Classe	Descrição	Instâncias
TipoDeArtefato	Um elemento de categoriza ou estereotipa o artefato	Plano de teste Código Fonte
Artefato	Um produto qualquer gerado durante uma instância de processo	Código fonte java da classe Pessoa
Versão	Uma versão de um dado artefato	Versão 1.3 do código fonte Java da classe Pessoa
Modificação	Uma descrição de uma modificação feita por uma pessoa em uma dada versão de artefato	-
Pessoa	Uma pessoa qualquer que participe do projeto	-
Configuração	Um grupo de versões de artefatos que representam um release de um dado produto.	-
Projeto	Um projeto de desenvolvimento de software qualquer	-

O próximo capítulo apresenta um protótipo inicial do serviço de coordenação de processos, como estudo de caso é apresentado a modelagem de um processo simples o SUP 4, processo para resolução de problemas encontrado na categoria suporte do SPICE.

CAPÍTULO 8

UM PROTÓTIPO DO SERVIÇO

Neste capítulo, descreve-se um protótipo inicial do serviço de coordenação de processos como estudo de caso modelaremos o processo para solução de problemas em projetos (SUP 4) *Perform Problem Resolution*, encontrado na categoria de processos Suporte do modelo SPICE (1993), após apresentadas a modelagem do processo, é apresentado como o processo é definido no ambiente, configurado e executado.

8.1.- Definição do Processo

Durante a execução de um processo de construção de software, muitos fatos prejudiciais ao projeto podem ocorrer, sejam eles previstos pela gerência ou não. De qualquer maneira atitudes e ações devem ser tomadas para que estes fatos não venham a prejudicar o andamento normal do projeto, principalmente aqueles que podem causar o comprometimento da qualidade e do custo.

É chamado de Ocorrência a qualquer evento ou fato que deva ser considerado como um potencial elemento de risco para o projeto e que necessite de avaliação pelo grupo responsável por tal tarefa. Uma ocorrência pode ser, por exemplo, uma inconsistência identificada durante uma revisão de análise entre, o modelo de classe/objeto desenvolvido e o documento de requisitos de software. Em função desta inconsistência e do fato detectado, alguma ação de modificação deve ser tomada para corrigir esta inconsistência.

Alguns trabalhos apresentam a relevância do tratamento de problemas em projetos para um bom gerenciamento. O modelo SPICE (SPICE, 1993) apresenta um processo relacionado a problemas e apresenta o processo SUP 4 – “Perform Problem Resolution” (categoria SUP- Suporte). O projeto “Capability Maturity model Integration” – CMMI (SEI, 2000) apresenta dentro de suas áreas de processos, vários componentes relacionados a problemas como o gerenciamento de risco (categoria gerência de projeto), análise de decisão e resolução (categoria suporte) e análise causal e resolução (categoria suporte).

8.1.1.- Propósito

O processo de solução de problemas deve garantir a análise e a correção de problemas encontrados dentro do projeto por qualquer pessoa envolvida.

8.1.2.- Atividades do Processo

O processo de solução de problemas deve ser iniciado a partir do momento em que qualquer pessoa envolvida no projeto encontre um problema, para tanto alguns procedimentos devem ser seguidos:

- **Abrir Ocorrência** - O “Relator”, qualquer pessoa envolvida no projeto e que encontra um problema, deve preencher um relatório de ocorrência, fornecendo informações referentes ao problema;
- **Analisar Ocorrência** - O analista, após ser notificado sobre a ocorrência deve verificar o correto preenchimento das informações fornecidas pelo relator e assim analisar a ocorrência. Uma vez analisada, o analista deve fornecer/confirmar informações sobre o problema verificar o possível impacto, o produto a ser corrigido e se a ocorrência procede, se sim o analista deve encaminhar a solução e um solucionador deve ser notificado sobre o problema, se a ocorrência for de alto risco ao projeto, a gerência de projeto deve ser notificada, se a ocorrência não proceder, deve então, ser descartada;
- **Solução da Ocorrência** - O solucionador de ocorrência, após ser notificado deve solucionar o problema descrito no relatório de ocorrência e assim fornecer as informações da solução, solução deve ser encaminhada para revisão e validação da solução; e
- **Revisão da Ocorrência** - O revisor deve revisar as informações fornecidas e compará-las com o critério de fechamento, fechando-a se os critérios forem atendidos, senão, a ocorrência deve permanecer em aberto indicando a necessidade de re-trabalho, o solucionador deve então ser notificado do re-trabalho.

As tarefas, entradas e saídas identificadas para o processo de solução de problemas são apresentadas na Tabela 8.1.

TABELA 8.1.- Relação de tarefas, entradas e saídas do processo de solução de problemas.

Tarefa	Entradas	Saídas	Responsável
Relatar Problema		Relatório de Problema	Relator de Ocorrência
Analisar Problema	Relatório de Problema	Relatório de Análise	Analista de Ocorrência
Notificar Gerência	Relatório de Análise		Agente Mensageiro
Solucionar Problema	Relatório de Análise ou Relatório de Revisão	Relatório de Solução	Solucionador de Ocorrência
Revisar Solução	Relatório de Solução	Relatório de Revisão	Revisor de Solução de Ocorrência

8.2.- Modelagem do Processo

A PML escolhida para modelagem do processo foi a E3 (Jaccheri et. al., 1998), esta PML tem suas raízes baseadas na orientação a objetos, diversas características como herança e associações são empregadas nesta PML. Um conjunto de elementos (tarefas, papéis, dados e ferramentas) e associações (precedência, feedback, responsável, entrada, saída, etc) com semânticas próprias são utilizadas na modelagem de processos.

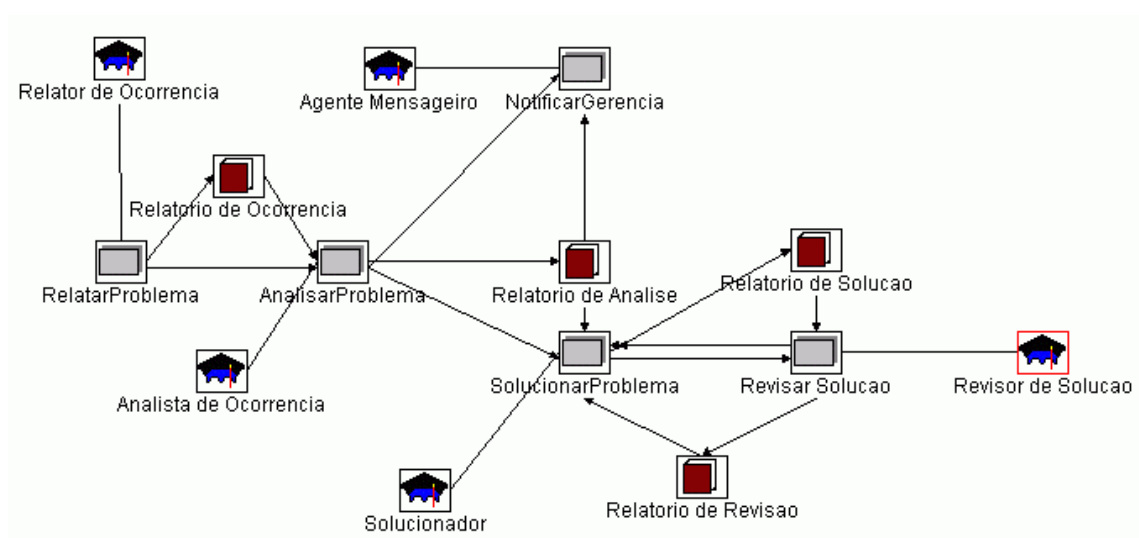


FIGURA 8.1.- Modelo de Processo do Processo de Solução de Problemas.

8.3.- Configuração do Modelo de Processo no e-WebProject

O processo é definido através da utilização da ferramenta PDE “Process Definition Environment” (Cereja Jr et. al., 2003a), conforme os passos a seguir:

1. Definição do Processo
2. Definição das tarefa do processo
3. Configuração de papéis das tarefas
4. Configuração de entradas e saídas

A Figura 8.2, apresenta um navegador de processos, mostrando o processo definido e suas respectivas tarefas.

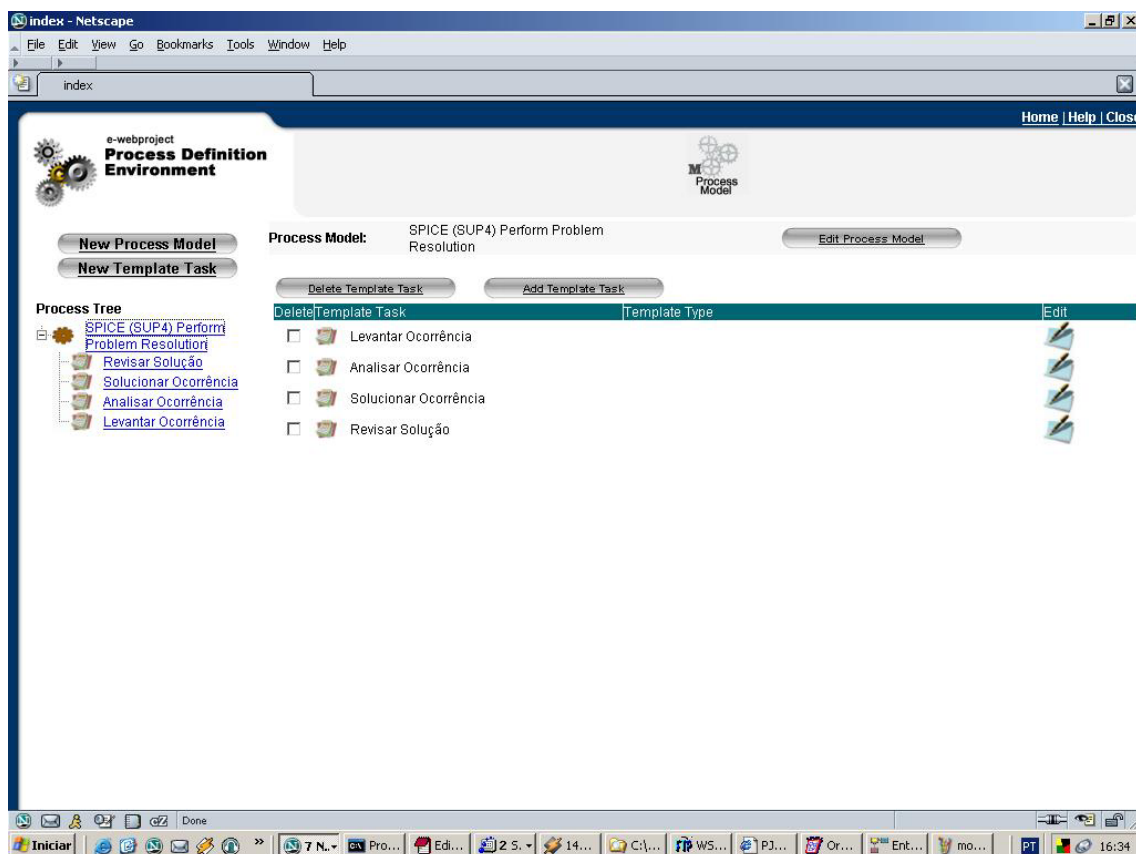


FIGURA 8.2.- Navegador de Modelos de Processos da ferramenta PDE.

A Figura 8.3 apresenta as propriedades de uma tarefa do processo.

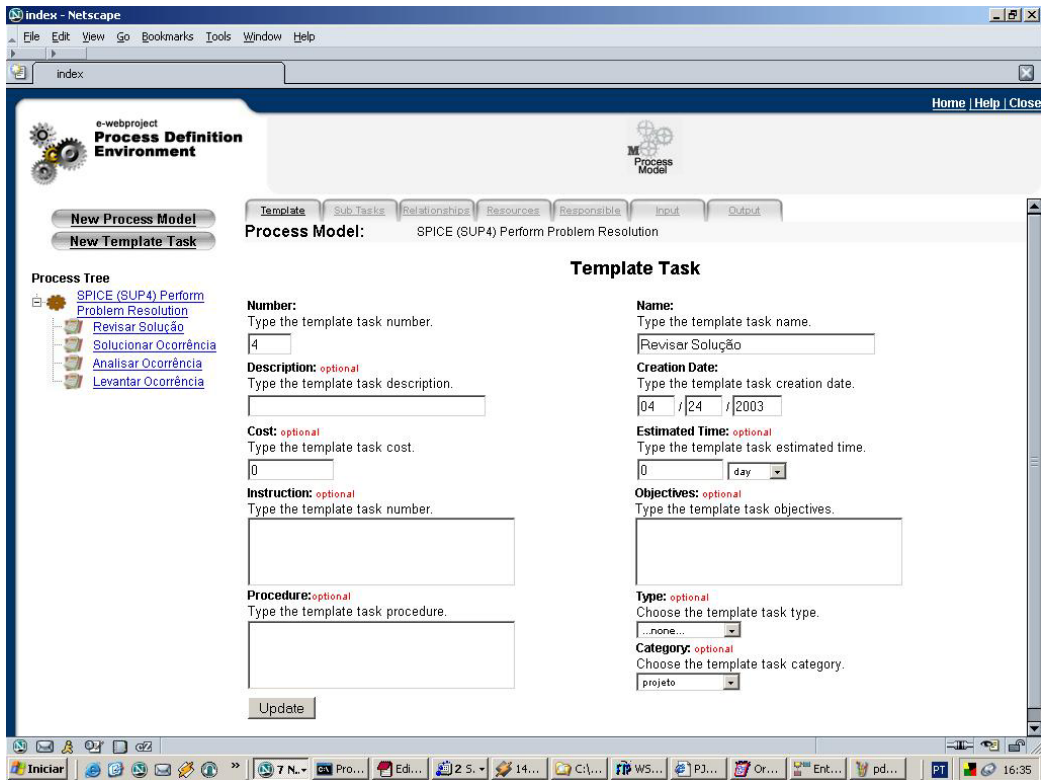


FIGURA 8.3.- Propriedades de uma tarefa do Processo.

A Figura 8.4 apresenta a configuração dos papéis de uma tarefa.

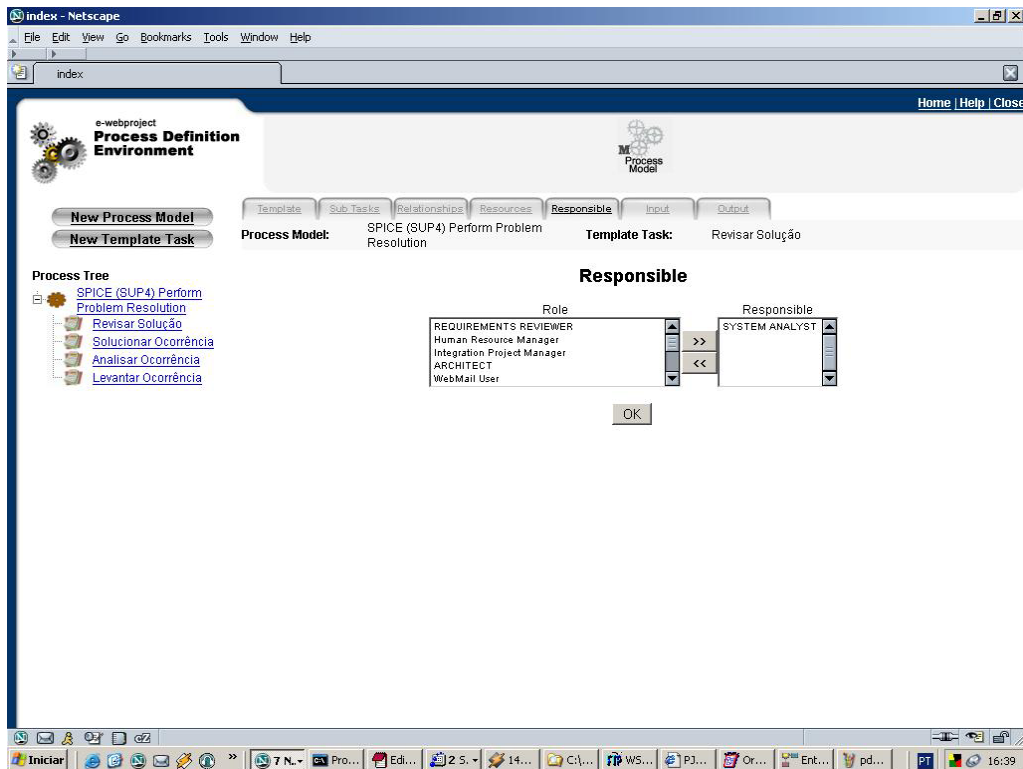


FIGURA 8.4.- Configuração de papéis responsáveis de uma tarefa.

A Figura 8.5 mostra a definição das entradas de uma tarefa.

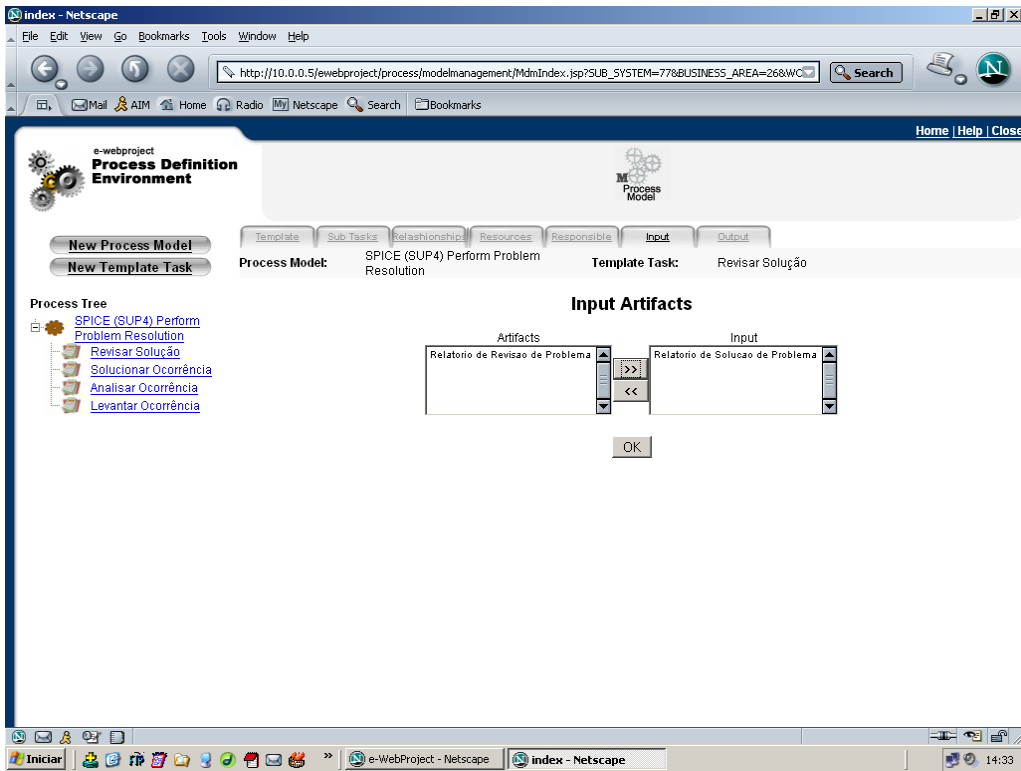


FIGURA 8.5.- Configuração de artefatos de entrada de uma tarefa.

A Figura 8.6 mostra a configuração dos artefatos de saída de uma tarefa.

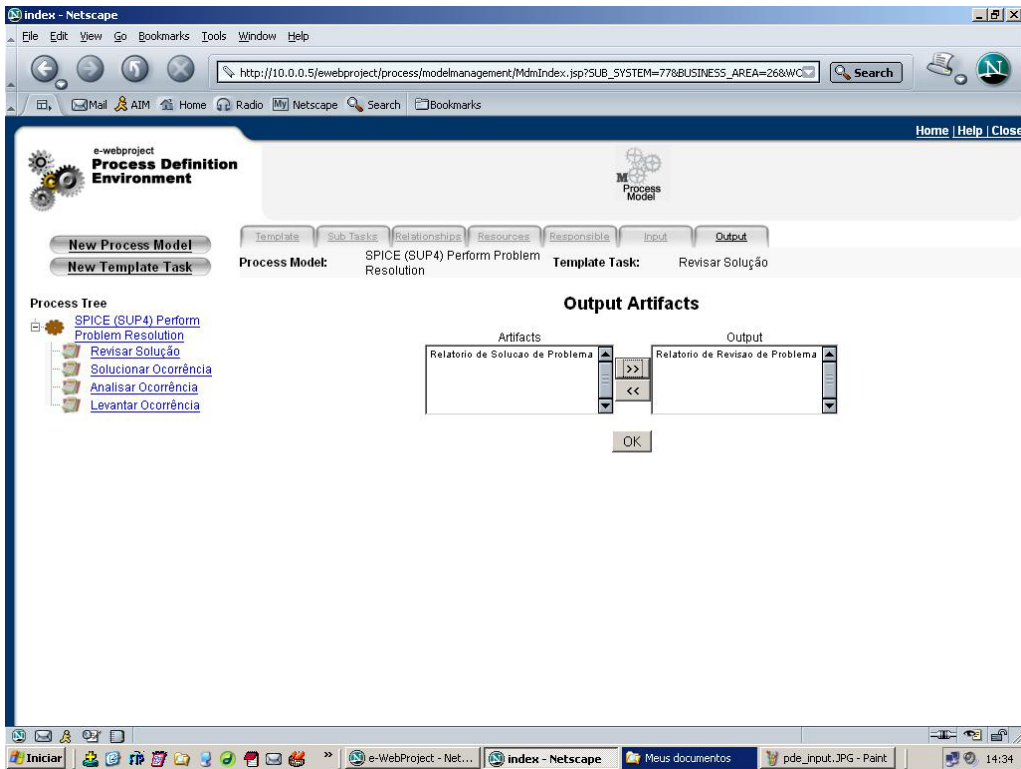


FIGURA 8.6.- Configuração de artefatos de saída de um processo.

8.4.- Configuração de uma instância de modelo de processo no e-WebProject

Um modelo pode ser instanciado no ambiente de duas formas, automática ou manual, o gerente de processos possui uma interface pelo qual ele seleciona os modelos de processos disponíveis, o e-WebProject apresenta um formulário, onde é apresentado as opções de configuração de uma instância do modelo de processo. Este formulário é apresentado na Figura 8.7.

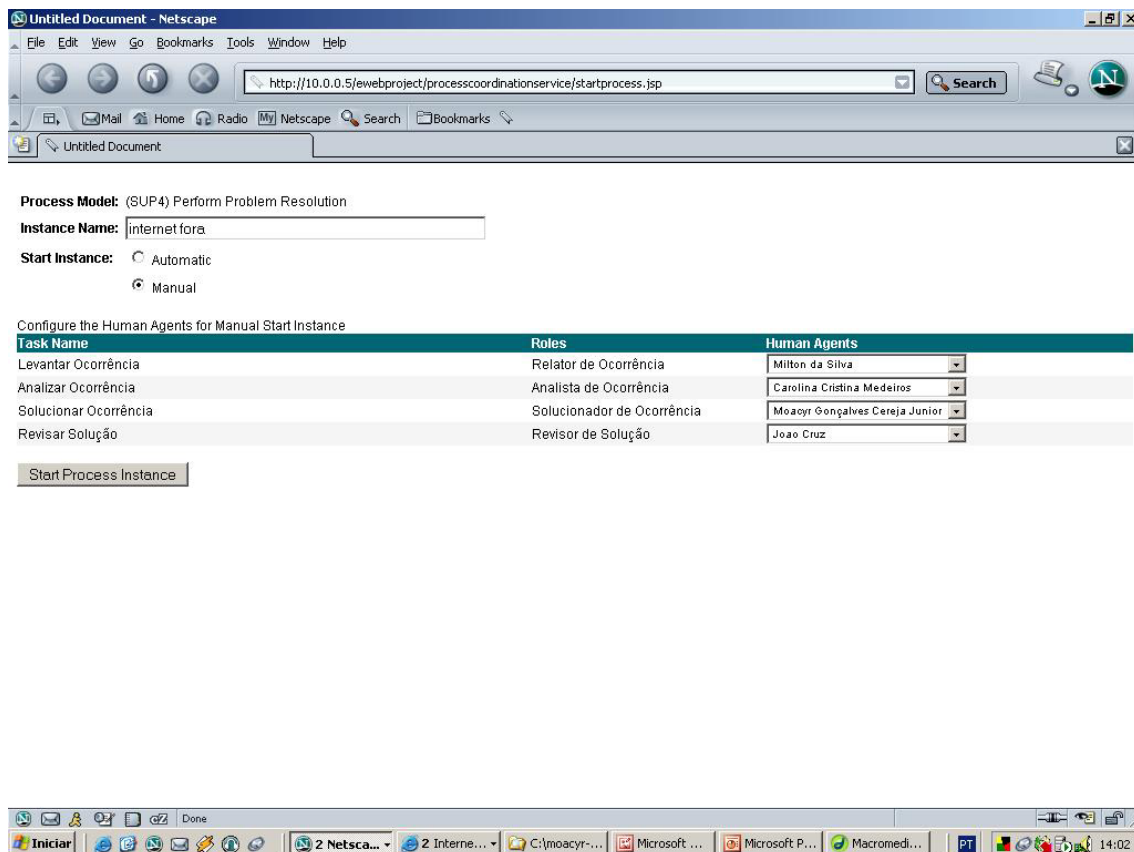


FIGURA 8.7.- Configuração de um instância de modelo de processo.

8.5. Alocação de uma tarefa de processo para um executor no ambiente

Cada usuário do e-WebProject, possui um espaço de trabalho eletrônico, onde ele tem acesso a ferramentas para o trabalho cooperativo, visualização de notícias e uma lista de tarefas próprias para aquele usuário estar executando durante um projeto de software. Esta lista de tarefas apresenta os estados das tarefas e um “link” para um tela de propriedades da tarefa, onde o usuário pode visualizar instruções, apontar tempos e visualizar gráficos de gantt pessoais.

A Figura 8.8, mostra o espaço de trabalho de um usuário com uma tarefa de uma instância do processo SUP 4, alocada para um usuário específico.

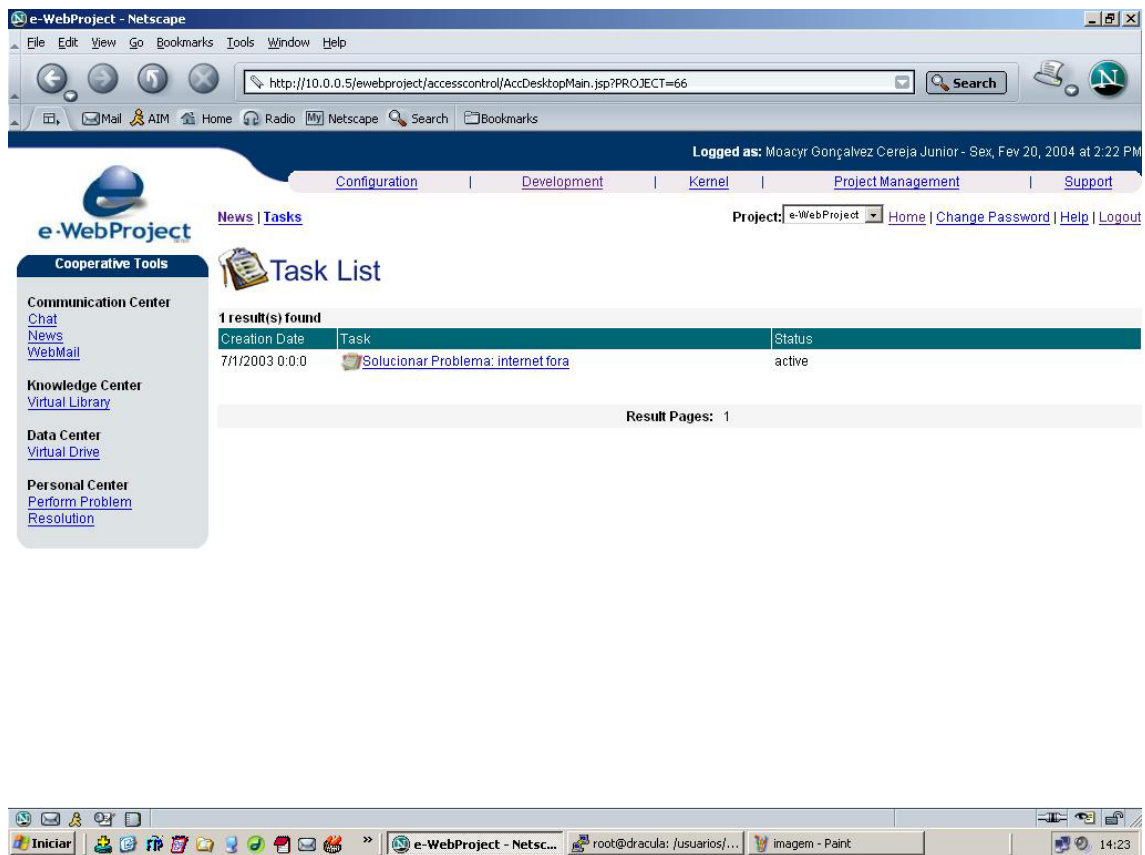


FIGURA 8.8.- Espaço de trabalho de um usuário do e-WebProject

Não foram apresentadas telas do repositório de artefatos porque o mesmo ainda encontra-se em fase de desenvolvimento, deste modo, não foi possível o seu término em tempo hábil para a publicação desta dissertação de mestrado.

CAPÍTULO 9

CONSIDERAÇÕES FINAIS

Procurou-se, com este trabalho de pesquisa, aprofundar o conhecimento do assunto processos de software. Para tanto, foram pesquisados, temas que englobam principalmente as atividades de definição e automação dos processos. Como objetivo principal, foi dada ênfase à proposta de um serviço de coordenação de processos integrado ao ambiente de engenharia de software centrado em processo e-WebProject (Sant'Anna 2000) (Sant'Anna et. al., 2002)..

Com respeito à definição de processos, o tema “Modelagem de Processos” teve um grande enfoque, onde foram apresentadas definições, características de processos de software e técnicas de modelagem, cita-se principalmente o assunto “Linguagens de Modelagem de Processos”, ao qual foi muito explorado neste trabalho, fornecendo base para alcançar o objetivo proposto para esta dissertação. A exploração do tema forneceu subsídios ao entendimento do elemento processo de software. Além disso, a atividade de formalização de processos, item central para a efetivação de qualquer esforço de melhoria de processos, tornou-se menos árdua com as pesquisas realizadas sobre este assunto.

Quanto à automação de processos de software, esse tema foi largamente explorado ao longo do desenvolvimento do trabalho. Procurou-se agregar elementos ao ambiente e-WebProject que facilitassem a automação dos processos. Deste modo, foram abordados assuntos relacionados à tecnologia de automação de processos. A automação de processos é uma característica essencial de ambientes de engenharia de software centrados em processos (PSEE) como o e-WebProject.

Com base nas características apresentadas por Sant'Anna (2000, 2002), foram apresentados os conceitos que norteiam o ambiente e-WebProject de modo a

descrever o domínio em que se insere este trabalho, guiando o desenvolvimento do serviço proposto.

Foram identificadas duas tecnologias potenciais aplicadas ao desenvolvimento do serviço de coordenação de processos: a tecnologia de fluxo de trabalho *workflow* e a tecnologia de agentes.

A tecnologia de *workflow* foi explorada devido as suas características primárias de apoio à automação de processos orientados por humanos, que é o caso dos processos de software. A tecnologia de agentes, possibilita a construção de sistemas com uma certa autonomia e inteligência. Essas características são essenciais ao serviço proposto, que tem a função de coordenar e executar os processos ao longo do tempo.

Uma vez exploradas as técnicas e tecnologias descritas, foi proposto um serviço de coordenação de processos composto por três módulos: o ambiente de definição de processos, a máquina de processos e um repositório de artefatos.

O Ambiente de Definição de Processos tem como função o apoio à definição de processos de software no ambiente e-WebProject e foi desenvolvido com base nas técnicas de modelagem de processos pesquisadas.

Verificou-se que a arquitetura de desenvolvimento do e-WebProject baseada na tecnologia J2EE não atendia aos requisitos de autonomia propostos para o serviço. Desse modo, foi agregado à arquitetura um contêiner de agentes que serviu como meio para o desenvolvimento da máquina de processos, módulo responsável pelo apoio à execução dos processos.

O repositório de artefatos tem como função controlar os artefatos produzidos durante a execução dos processos, isto é, controle de versão e configuração; e indicação de informação para controle da execução dos processos, onde a

informação de conclusão ou não de um artefato pode servir como restrição para conclusão ou não de tarefas dos processos.

Por último, foi apresentado um protótipo inicial do serviço, onde foi utilizado um Estudo de Caso, a modelagem do processo de solução de problemas em projeto, para mostrar o funcionamento do protótipo.

9.1.- Conclusões

Vale salientar que antes do desenvolvimento desse serviço, era necessário, para automação de um processo, o desenvolvimento de aplicativos específicos. Uma vez desenvolvidos, estes aplicativos eram integrados ao e-WebProject. Esta abordagem inflexível trazia alguns transtornos, já que era necessário novo desenvolvimento a cada alteração no modelo de processo.

Com o serviço de coordenação de processo apresentado neste trabalho, a atividade de automação de processos foi minimizada. A definição de um novo processo no ambiente foi facilitada em muito e devido aos novos elementos agregados ao ambiente, constatou-se uma diminuição do esforço para se automatizar um processo.

9.2.- Contribuições

As principais contribuições observadas com a conclusão deste trabalho encontram-se descritas a seguir:

- As pesquisas relacionadas com a modelagem de processos proporcionaram definições mais claras dos processos de software adotados por uma organização, isto forneceu subsídios para o andamento de outros trabalhos relacionados ao tema e-WebProject (Borrego Filho et. al., 2002a, 2002b, 2003a, 2003b) (Genvigir et. al., 2002, 2003);

- A modelagem da máquina de processos fornece uma base para o efetivo apoio à execução de processos de software, e é considerada a principal contribuição deste trabalho de pesquisa;
- A construção de um protótipo inicial do serviço de coordenação de processos pode ser utilizada para experimentações e estudos sobre a eficácia da automação de processos;
- Deixa-se um bom legado, proporcionando ainda uma fonte de pesquisa que leva a evolução do ambiente e-WebProject;
- Procurou-se apresentar à comunidade de Engenharia de Software os conceitos que norteiam o desenvolvimento do serviço de coordenação de processos. Isso pode ser observado, através da publicação de quatro artigos: três em congressos internacionais e um em workshop nacional estas referências podem ser vistas em (Cereja Jr. et. al., 2002a, 2002b, 2003a, 2003b).

Apesar das contribuições apresentadas, esse trabalho é uma proposta inicial que pode ser evoluído. Desse modo, procura-se identificar algumas perspectivas de trabalhos futuros que podem ser aplicados ao serviço.

9.3.- Perspectivas de Trabalhos Futuros

A conclusão deste trabalho abre campo para novas pesquisas que tem como intuito a evolução do ambiente e-WebProject. Propõem-se como trabalhos futuros:

- Agregar novos agentes com características de inteligência ao serviço de agentes definido para a máquina de processos. Agentes com comportamentos pró-ativos podem tornar-se um recurso interessante para prever falhas e interagir com os usuários do ambiente de forma mais eficiente;

- Como apresentado no Capítulo 4, o serviço de coordenação de processos não apóia a fase de avaliação e proposta de melhorias. Recomenda-se evoluir o serviço para o apoio a essa fase, já que fornece subsídios para a melhoria de processos. A pesquisa sobre métricas de software pode auxiliar este fim;
- O desenvolvimento de uma ferramenta CASE, que pode ser agregada ao ambiente de definição de processos, que apóie a definição de modelos com apoio de uma PML ou mais, facilitando assim, o trabalho do modelador de processos;
- O custo da automação de um processo pode ser alto, a execução de processos envolve a alocação de recursos humanos e o mesmo pode conter falhas, o que pode ocasionar em defeitos no produto final o que pode acarretar em mais custos. Para minimizar esses obstáculos podem-se aplicar recursos de simulação de processos, procurando assim, detectar defeitos antes do mesmo entrar em operação;
- Integração do e-WebProject com outras ferramentas de modelagem existentes no mercado.

REFERÊNCIAS BIBLIOGRÁFICAS

Acuña, S. T. Software process modelling. In: World Multiconference on Systemics, Cybernetics and Informatics, 5., 22-25 July 2001, Orlando, Florida, USA. **Proceedings...** Disponível em: <<http://www.unse.edu.ar/congres/jornadas/silviac/documentos/congres1.pdf>>. Acesso em: 26 ago. 2003

Ambriola, V.; Conardi, R.; Fungetta, A. Assessing process centered software engineering environments. **Transactions on Software Engineering and Methodology**, v. 6, n. 3, p. 283-328, 1997.

Balzer, R.; Dyer, D.; Morgenstern, M.; Neeches, R. Specifications Based Computing Environments. In: National Conference on Artificial Intelligence, 3. 1983, Washington. **Proceedings...** Washington: AAAI, 1983.

Ben-Shaul, Israel Z.; Kaiser, Gail E. **A paradigm for decentralized process modeling and its realization in the Oz environment**. New York: Columbia University, 1994. p.179-188.

Ben-Shaul I. Z.; Kaiser G. E. **Process support for synchronous groupware activities**. New York: Columbia University, 1995. (Technical Report CUCS-002-95).

Borrego Filho, L. F.; Sant'Anna, N.; Cereja Jr., M. G.; Genvigir, E. C. Análise e Modelagem para Automação de Processos de Gerenciamento de Projetos. In: WORCAP, 2., 2002, São José dos Campos. **Anais...** São José dos Campos: Instituto Nacional de Pesquisas Espaciais, 2002.

Borrego Filho, L.F.; Sant'Anna, N.; Cereja Junior, M.G.; Genvigir, E.C.; Luque, L. Uma aplicação para automatização dos processos de gerenciamento de projetos suportada por ambientes de engenharia de software. In: Congresso Internacional de Tecnologia de Software, 14., jun. 2003, Curitiba. **Anais...** Curitiba: [S.n.], 2003a.

Borrego Filho, L.F.; Sant'Anna, N.; Cereja Júnior, M.G.; Genvigir, E.C. Proposta de arquitetura para apoio, suporte e automação de processos de gerenciamento de projetos. In: Worcap, 3., 2003, São José dos Campos. **Anais...** São José dos Campos: Instituto Nacional de Pesquisas Espaciais, 2003b.

Borrego Filho, L.F.; Sant'Anna, N.; Cereja Junior, M. G.; Luque, L.; Casilo, B. H. Uma abordagem para processos de gerenciamento do tempo e dos recursos humanos apoiados por ambientes integrados de engenharia de software. In: Simpósio Internacional de Melhoria de Processos de Software – SIMPROS, , 2002, Recife. **Anais...** Recife: CenPra, 2002b.

Boudier, G.; Gallo, F.; Minot, R.; Thomas, I. An Overview of PCTE and PCTE+. In: ACM SIGSOFT Software Engineering Symposium an Practical Software Engenering Environments., Dec. 1988. **Proceedings...** Boston: ACM, 1988. p.248–257.

Bourne, S. R. The UNIX shell. **The Bell System Technical Journal**, v.57, n.6, p.1971–1990, July-Aug. 1978.

Cagan, M. R. The HP SoftBench environment: an architecture for a new generation of software tools. **Hewlett-Packard Journal**, p.36–47, June 1990.

Cereja Júnior, M.G.; Sant'Anna N.; Borrego Filho, L.F.; Luque L.; Casillo, B.H. Sant'Anna N. Modelando um processo de solução de problemas: UML e PML: uma exploração de abordagens para a modelagem de processos. In: Simpósio Internacional de Melhoria de Processo de Software,4., 10-13 set. 2002. **Anais...** Recife: CenPra, 2002a.

Cereja Júnior, M.G.; Sant'Anna N.; Borrego Filho, L.F.; Luque L.; Casillo, B.H. Sant'Anna, N. UML e PML: uma exploração de abordagens para a modelagem de processos. In: Workshop dos Cursos de Computação Aplicada do INPE, 2., nov. 2002, São José dos Campos. **Anais...** São José dos Campos: Instituto Nacional de Pesquisas Espaciais, 2002b.

Cereja Junior, M.G.; Sant'Anna N.; Borrego Filho, L.F.; Luque L.; Casillo, B.H.; Tavares, R. P. Process Definition Environment: um ambiente para apoio à definição de processos de software no ambiente e-WebProject. In: Congresso Internacional de Tecnologia de Software,14., 24-27 jun. 2003, Curitiba. **Anais...** Curitiba: [S. n.], 2003a. p. 42-58.

Cereja Junior, M.G.; Sant'Anna N.; Casillo, B.H.; Tavares, R.P.; Oliveira Pinto, N.L.; Cunha Chagas, H.L. Uma abordagem para automação de processos de software no ambiente de engenharia de software e-WebProject. In: Simpósio Internacional de Melhoria de Processo de Software, 5., nov. 2003, Recife. **Anais...** Recife: [S. n.], 2003b.

Christie Alan M. **Process-centered development environments: an exploration of issues.** Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, 1993. (CMU/SEI-93-TR-4).

Conallen J. **Building web applications with UML.** Massachusetts: Addison Wesley, 2000.

Conradi, R.; Fernstrom, C.; Fuggeta, A.; Snowdon, B. Towards a reference framework for process concepts. In: European Workshop on Software Process Technology, 2., 1992, Berlim **Proceedings...** London: Springer- Verlag, 1992.

Conradi, R.; Jaccheri, M.L. Process modelling languages. In: Derniame, J.C.; Kaba, B.A.; Wastell, D. (eds). **Software process: principles, methodology, and technology**. Berlin: Springer Verlag, 1998.

Cruz, Tadeu. **Workflow: a tecnologia que vai revolucionar processos**. 2.ed. São Paulo: Atlas, 1998.

Cugola G.; Ghezzi, C. Software Processes: a retrospective and a path to the future. In: International Conference on Software Process, 5., 14-17 June 1998, Lisle. **Proceeding...** [S.l.]: [S.n.], 1998.

Derniame J.C.; Kaba, B.A.; Wastell, D.G. **Software process: principles, methodology, technology**. Berlin: Springer Verlag, 1999. (Lecture Notes in Computer Science, v.1500).

Dowson, M.; Nejme, B.; Riddle, W. Fundamental software process concepts. In: European Workshop on Software Process Modeling, 1., Apr. 1991, Milan. **Proceedings...** [S.l.]: [S.n.], , 1991.

El Emam, K.; Briand L. Costs and benefits of software process improvement. In: **Better software practice for business benefit: principles and experience**. IEEE, 1999. Disponível em: <<http://citeseer.nj.nec.com/elemam97costs.html>> . Acesso em: 26 ago. 2003.

Ellmer E. **Process centered software engineering environments as the next-generation of CASE tools**. Feb. 1995. Disponível em: <<http://www.ifs.univie.ac.at>>. Acesso em: 26 ago. 2003.

Fuggetta. A. Software process: a roadmap. In: Finkelstein, A. (ed). **The future of software engineering**. [S.l.]: ACM Press, 2000. Disponível em: <<http://www.cs.ucl.ac.uk/staff/A.Finkelstein/fose/finalfinkelstein.pdf#search=%22Software%20process%20roadmap%20The%20future%20of%20software%20engineering%22>>. Acesso em: 20 ago. 2003.

Franch X.; Ribó, J.M. Using UML for modelling the static part of a software process. In: UML99 - Beyond the Standard International Conference, 2., 28-30 Oct. 1999. Fort Collins CO. **Proceedings...** [S.l.]: [S.n.], 1999a

Franch, X.; Ribó, J.M. **PROMENADE: a modular approach to software process modeling and enactment**. 1999b. Disponível em: <<http://citeseer.nj.nec.com/270359.html>>. Acesso em: 20 ago. 2003.

Feiler, P.H.; Humphrey, W.S. **Software process development and enactment: concepts and denitions**. Pittsburgh: Carnegie Mellon University, 1991. (CMU/SEI-92-TR-004)

Feldman, S. Make - a program to maintain programs. **Software Practice and Experience**, v.9, n.3, p.255–265, Mar. 1979.

Foundation for Intelligent Physical Agents (FIPA), **FIPA Abstract Architecture Specification**. 2002. Disponível em: <<http://www.fipa.org/>>. Acesso em 21 ago. 2003.

Garg, P.K.; Jazayeri, M. **Process-centered software engineering environments**. Los Alamitos: IEEE Computer Society Press, 1996.

Genvigir, E.C.; Sant'Anna, N.; Borrego Filho, L.F.; Cereja Junior, M.G. Uma abordagem para os processos da engenharia de requisitos. In: WORCAP, 2., 2002, São José dos Campos. **Anais...** São José dos Campos: Instituto Nacional de Pesquisas Espaciais, 2002.

Genvigir, E.C.; Sant'Anna, N.; Borrego Filho, L.F.; Cereja Junior, M.G.; Casilo, B.H. Modelando processos de software através do UPM: modelo de processo unificado. In: Congresso Internacional de Tecnologia de Software, 14., jun. 2003, Curitiba. **Anais...** [S.l.]: [S.n.], 2003.

Georgakopoulos D.; Hornick M. An overview of workflow management: from process modeling to workflow automation infrastructure. **Distributed and Parallel Databases**, v.3, p.119-153, 1995.

GiMENES, I.M.S. O processo de engenharia de software: ambientes e formalismos. In: Jornada de Atualização em Informática, 13., set. 1994, Caxambu. **Anais...** [S.l.]: JAI, 1994. p.02_42.

GiMENES, I.M.S. **ExpSEE**: um ambiente experimental de engenharia de software orientado a processos. Maringá: Universidade Estadual de Maringá, Departamento de Informática, 2000. (Relatório de Projeto).

Green, C.; Luckam, D.; Balzer, R.; Cheatham, T.; Rich, C. **Report on a Knowledge Based Software Assistant**. [S.l.]: Kestrel Intitute, June 1983. (Technical Report KES.U.83.2).

Hagen C.; Alonso G. Beyond the black box: event-based inter process communication in process support systems. In: International Conference on Distributed Computing Systems, 19., 1999. **Proceedings...** [S.l.]: IEEE, 1999.

Herbsleb J.; Carleton A.; Rozum J.; Siegel J.; Zubrow D. **Benefits of CMM-based software process improvement**: initial results. Pittsburg: Carnegie Mellon Univeristy, 1994. (Technical Report, CMU/SE-94-TR-013). Disponível em: <

<http://www.sei.cmu.edu/publications/documents/94.reports/94.tr.013.html>>. Acesso em: 26 ago. 2003.

Hollingsworth, D. **The workflow reference model**. [S.l.]: Workflow Management Coalition Specification 1995. (TC00-1003). Disponível em: <<http://www.wfmc.org/standards/docs/tc003v11.pdf>>. Acesso em 25 ago. 2003.

Humphrey Watts S.; Kellner Marc, I. **Software process modeling: principles of entity process models**. Pittsburgh: Carnegie Mellon University, 1989. (CMU/SEI-89-TR-2).

Humphrey W. S. **A discipline for software engineering**. [S.l.]: Addison-Wesley Publishing, 1995.

Ingalls, D. The smalltalk-76 programming system, design and implementation. In: Annual ACM Symposium on Principles of Programming Languages, 5., jan. 1978. **Proceedings...** [S.l.]: Association of Computing Machinery, 1978. p.9-17.

International Standard Organization. **ISO/IEC 12207: Standard Software Life Cycle Processes**, 1995.

Jaccheri Maria L; Picco Gian P.; Lago, P. Eliciting software process models with the E³ language. **ACM Transactions on Software Engineering and Methodology**, v.7, n 4, p.368-410, Oct. 1998, 368-410.

Jaccheri Maria L.; Baldi M.; Divitini M. Evaluating the requirements of software process modeling languages and systems. In: ACM (ed.). **Process support for distributed team-based software development (PDTSD99)**. Orlando, Florida: PDTSD, 1999. p.570-578.

Jacobson I.; Booch G.; Rumbaugh J. **The unified modeling language user guide**. Massachusetts: Addison-Wesley Publishing, 1999a.

Jacobson I.; Booch G.; Rumbaugh J. **The unified software development process**. Massachusetts: Addison-Wesley Publishing, 1999b.

JADE - **Java Agent Development Framework**. 2003. Disponível em: <<http://jade.cselt.it>>. Acesso em: 23 ago. 2003.

Kadia, R. Issues encountered in building a flexible software development environment: lessons from the arcadia project. In: ACM SIGSOFT Symposium on Software Development Environments, Washington, 1992. **Proceedings...** [S.l.]: ACM, 1992.

Kaiser, G.E.; Feiler, P.H. An Architecture for Intelligent Assistance in Software Development. In: International Conference on Software Engineering, 9., 1987, Monterey, Califórnia. **Proceedings...** [S.l.]: [S.n.], 1987. p.180–188.

Kellner, M.I.; Hansen, G.A. **Software process modeling**. Pittsburgh: Carnegie Mellon University, 1988. (CMU/SEI-88-TR-9).

Kontonya, G.; Sommerville, I. **Requirements engineering : process and techniques**. Chichester: John Wiley & Sons, 2000.

Lonchamp, J; A structured conceptual and terminological framework for software process engineering. In: International Conference on the Software Process - Continuous Software Process Improvement, 2., 1993. **Proceedings...** [S.l.]: [S.n.], 1993.

Madhavji, N.H. The process cycle. **IEEE Software Engineering Journal**, v.6, n.5, p.234-242, 1991.

Ministério da Ciência e Tecnologia. **Qualidade e produtividade no setor de software, glossário de termos da qualidade**. 1999. Disponível em: <<http://www.mct.gov.br/Temas/info/Dsi/qualid99/99anexo2.htm>>. Acesso em: 22 ago. 2003.

Moraes, S. W. **Um ambiente expert para apoio ao desenvolvimento de software**. Porto Alegre: [S.n.], 1997. 133p.

Mühlen Z.M.; Becker J. Workflow process definition language: development and directions of a meta-language for workflow process. In: KnowTech Forum, 1., 17-19 Sept. 1999. **Proceedings...** [S.l.]: [S.n.], 1999.

Nwana, Hyacinth S. Software agents: an overview. **Knowledge Engineering Review**, v.11, n.3, p.1-40, Sept. 1996.

Object Management Group. **MOF Meta Object Facility specification**. Version 1.4. [S.l.]: OMG, Nov. 2001.

Object Management Group. **The Unified Process Model (UPM)**: initial submission. [S.l.]: OMG, May 2000.

Odell, J.; Parunak, H.V.D.; Bauer, B. Extending UML for Agents. In: Agent-Oriented Information Systems Workshop; National conference on Artificial Intelligence, 17., 2000. **Proceedings...** Disponível em: <<http://citeseer.nj.nec.com/odell00extending.html>>. Acesso em: 22 ago. 2003.

Oliveira, K.M. **Modelo para construção de ambientes de desenvolvimento orientados a domínio**. 1999. Tese (Doutorado) - Universidade Federal do Rio de Janeiro, 1999.

Osterweil, L. Software process are software too. In: International Conference on Software Engineering, 9., Mar.1987, Monterey, Califórnia. **Proceedings...** Califórnia: ACM, 1987.

Paula Filho, W.P. **Engenharia de software: fundamentos, métodos e padrões**. Rio de Janeiro: LTC, 2000.

Project Management Institute. **A guide to the project management body of Knowledge**. Charlotte,NC: PMI, 2000.

Reis, C.A.L. **APSEE: uma abordagem baseada em conhecimento para gerência de processos de software evolutivos**. Porto Alegre: Universidade Federal do Rio Grande do Sul, jul. 2001. (Proposta de Tese).

Reis, R.Q. **Uma proposta de suporte ao desenvolvimento cooperativo de software no ambiente PROSOFT**. Porto Alegre: [S.n.], 1998. 177p.

Reis, S. Connecting tools using message passing in the FIELD programming environment. **IEEE Software**, v.7, n.4, p.276–284, July 1990.

Ritchie, T. UNIX: a time sharing operating system. **Communications of the ACM**, v.17, n.7, p.365–375, July 1974.

Sant'Anna N. **Um ambiente experimental para a engenharia de software**. São José dos Campos. (INPE-5540-TDI/528). 1993. Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 1993.

Sant'Anna, N. **Um ambiente integrado para apoio ao desenvolvimento e gestão de projetos de software para sistemas de controle de satélites**. São José dos Campos: Instituto Nacional de Pesquisas Espaciais, 2000. (INPE - 8306 - TDI/765).

Sant'Anna, N.; Cereja Junior, M.G.; Borrego Filho, L.F.; Luque L.; Casillo, B.H. e-WebProject: um ambiente integrado para o apoio ao desenvolvimento e gestão de projetos de software. In: Congresso Internacional de Tecnologia de Software: Qualidade e Produtividade no Gerenciamento de Projetos, 13., 19-21 jun. 2002, Curitiba. [S.l.]: [S.n.], 2002. p.163-174.

Schleicher A.; Westfechtel B.; Jäger D. **Modeling dynamic software process in UML**. Germany: RWTH, 1998. (Technical Report AIB 98-11).

Schleicher A. Formalizing UML-based process models using graphs transformations. In: ACM (ed.). **Proceedings of the International Workshop on Applications of Graph Transformations with Industrial Relevance**. [S.l.]: ACM, 2000. p.341-357.

Schleicher A.; Westfechtel B. Beyond stereotyping: metamodeling approaches for the UML. In: Hawaiian International Conference on System Sciences, 34., 2001, Los Alamitos, CA. **Proceedings...** [S.l.]: IEEE Computer Society Press, 2001.

Smith, D.R.; Kotik, G.B.; Westfold, S.J. Research on Knowledge-Based Software Environments at Kestrel Institute. **IEEE Transactions on Software Engineering**, v.11, p.1278–1295, Nov. 1985.

Sousa, A.L.R.; Reis, C.A.L.; Reis, R.Q.; Pimenta, M.S.; Nunes, D.J. Analisando a interação de gerentes e desenvolvedores em ambientes de processo de software: classificação e exemplos. In: Jornadas Chilenas de Computação; Workshop de Engenharia de Software, 2001, Punta Arenas. **Proceedings...** [S.l.]: [S.n.], 2001

Software Process Improvement and Capability dEtermination. **Software process assessment – Part 2** : a model for process management. Version 1.00, 1993 SPICE

Software Engineering Institute (SEI). **Capability maturity model for software** Version 1.1. Pittsburgh: Software Engineering Institute, Feb. 1993a. (Technical Report CMU/SEI-93-TR-24).

Software Engineering Institute (SEI). **CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development**. Version 1.2. Pittsburgh, Pennsylvania, 2000. (CMU/SEI-2000-TR-031).

Taylor, R.N.; Belz, F.C.; Clarke, L.A.; Osterweil, L.; Selby, R.W.; Wileden, J.C.; Wolf, A.L.; Young, M. Foundations for the arcadia environment architecture. In: Henderson, P. (ed.). **Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments**. Boston: ACM, 1988. p.1–13.

Teitelman, W.; Masinter, L. The interlisp programming environment. In: Barstow, D.R.; Shrobe, H.E.; Sandelwall, E. (ed.). **Interactive Programming Environments**. New York: McGraw-Hill, 1984. p.3–18.

Travassos, G. H. **O modelo de integração de ferramentas da estação TABA**. 1994. Tese (Doutorado) Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1994.

Wang H.; Dongming X. Collaborative multi-agents for workflow management.
In: Hawaii International Conference on System Sciences, 34., 2001.
Proceedings... [S.l.]: [S.n.], 2001.

Weiss G. **Multiagent systems**: a modern approach to distributed modern
approach to artificial intelligence. Massachusetts: MIT Press, 1999.

APÊNDICE A

ALGUMAS ABORDAGENS PARA A MODELAGEM DE PROCESSOS

Neste apêndice são descritas algumas abordagens utilizadas para a modelagem de processos de software, dentre essas abordagens encontram-se a descrição das Linguagens de Modelagem de Processos (Process Modeling Languages PMLs): ProNet, E3, PROMENADE. A seguir, são apresentados os seus elementos; algumas notações gráficas; a apresentação do UPM (Unified Process Model) uma submissão inicial da OMG para a padronização da modelagem de processos de software; por último, uma abordagem para a utilização da UML como linguagem para a modelagem de processos.

A.1. A Linguagem de Modelagem Gráfica ProNet

A linguagem gráfica ProNet (Christie, 1993), é uma representante da primeira geração de PMLs. Dadas as várias definições sobre processos apresentadas na seção 1. Atividades se apresentam como um elemento central em qualquer modelo de processo, e como não poderia ser de forma diferente, elas assumem uma posição central na linguagem ProNet. Uma atividade somente pode ocorrer se certas condições de entrada são satisfeitas ou se certos produtos estão disponíveis. Como consequência de uma atividade, condições de saída podem ser mudadas de falso para verdadeiro (ou vice versa) e certos produtos podem ser gerados. Partindo do pré suposto que uma atividade muda o estado de um sistema, gerando e configurando condições de entrada necessárias para o disparo de outras atividades, conclui-se a linguagem ProNet é um modelo declarativo.

Cada atividade e suas condições de entrada e saída tem um forte relacionamento com elementos de uma regra em um sistema baseado em regras, e é por causa destas características que a linguagem torna a executabilidade de um processo possível.

A notação gráfica da linguagem ProNet pode ser mapeada para um conjunto de regras através do qual um processo pode ser executado. O modelo simbólico resultante pode ser usado para investigar características dinâmicas de um processo, ou, igualmente importante, para definir características de uma máquina de processos em um Ambiente de Engenharia Software Centrado em Processo.

A.1.1. Classes de Entidades da Linguagem ProNet

Diagramas ProNet são baseados em um modelo entidade relacionamento modificado, cujas entidades seguem uma das seguintes classes listadas a seguir:

- **Atividades:** são a espinha dorsal de um modelo de processo. Produtos e condições (definidos abaixo), fornecem condições para o início de uma atividade. Atividades são responsáveis por gerar produtos e condições de saída.
- **Produtos:** podem ser solicitados para o apoio a uma atividade ou ser produzidos por uma atividade. Produtos podem ser o resultado de alguma atividade interna do modelo (um código fonte, por exemplo) ou ser gerado fora dela.
- **Condições:** podem tanto ser solicitadas para iniciar uma atividade ou ser resultado de uma atividade (“revisão completada”, por exemplo) e tem valores VERDADEIRO ou FALSO. A existência ou não existência de um produto, agente, etc., pode ser equivalente a uma condição.
- **Composições:** são combinações booleanas de condições, produtos, agentes, etc.
- **Agentes:** são entidades específicas que executam atividades. Entidades humanas ou não humanas capazes de executar atividades

(desenvolvedor de software, compilador C, por exemplo) podem ser considerados agentes.

- **Roles:** são abstrações, uma super-classe do conceito de agente (revisor, editor, por exemplo). Se um modelo de processo é executado, o papel deve ser instanciado por um agente em tempo de execução.
- **Armazéns (Stores):** permitem a persistência de entidades instanciadas. Produtos, valores de condições e agentes podem ser depositados ou recuperados de armazéns.
- **Regras (Constraints):** são restrições impostas para a execução de uma atividade. Diferentemente das condições, as regras não possuem valores booleanos, mas podem refletir um guia de como as coisas são feitas (uma regra de garantia de qualidade como documentação de um código, por exemplo).

A maioria dos relacionamentos das entidades da linguagem ProNet são para atividades. Estes tipos de relacionamentos são mostrados nas tabelas A.1 e A.2 a seguir:

TABELA A.1 - Relacionamentos de Entrada.

Relacionamentos de Entrada	Relações de Entradas inversas
<i>Produto_A é produto de entrada para a atividade_A</i>	<i>Atividade_A tem como produto de entrada produto_A</i>
<i>Condição_A é condição de entrada para a atividade_A</i>	<i>Atividade_A tem como condição de entrada condição_A</i>
<i>Composição_A é composição de entrada para a atividade_A</i>	<i>Atividade_A tem como composição de entrada composição_A</i>
<i>Agente_A é agente de entrada para a atividade_A</i>	<i>Atividade_A tem como agente de entrada agente_A</i>
<i>Papel_A é papel de entrada para a atividade_A</i>	<i>Atividade_A tem como papel de entrada papel_A</i>
<i>Armazém_A é armazém de entrada para a atividade_A</i>	<i>Atividade_A tem como armazém de entrada armazém_A</i>

TABELA A.2 - Relacionamentos de Saída.

Relacionamentos de Saída	Relações de Saída Inversa
<i>Produto_A é produto de saída para a atividade_A</i>	<i>Atividade_A tem como produto de saída produto_A</i>
<i>Condição_A é condição de saída para a atividade_A</i>	<i>Atividade_A tem como condição de saída condição_A</i>
<i>Composição_A é composição de saída para a atividade_A</i>	<i>Atividade_A tem como composição de saída composição_A</i>
<i>Agente_A é agente de saída para a atividade_A</i>	<i>Atividade_A tem como agente de saída agente_A</i>
<i>Papel_A é papel de saída para a atividade_A</i>	<i>Atividade_A tem como papel de saída papel_A</i>
<i>Armazém_A é armazém de saída para a atividade_A</i>	<i>Atividade_A tem como armazém de saída armazém_A</i>

Nos diagramas de processo, todos os relacionamentos estão escritos em itálico. A informação contida no relacionamento, permite ao leitor, identificar a classe de entidade que está conectada à atividade. Um relacionamento (como “produto_A é produto de entrada para atividade_A”) possui um relacionamento inverso (“atividade_A tem como produto de entrada produto_A”).

Nos diagramas de processo, as atividades sempre podem ser identificadas já que o nome da entidade sempre está cercado por uma caixa ou retângulo sombreado. Em adicional, um ponto negro é colocado ao final de cada relacionamento. Por exemplo, no relacionamento “ABC é produto de entrada para XYZ”, o ponto apareceria próximo à caixa contento XYZ.

A.1.2. Notação Gráfica

Como mencionado anteriormente, a noção de atividade é central. Geralmente o objetivo de um processo de software é produzir um produto produtos de software. Assim, a função de algumas atividades é produzir produtos, outras são tomadas de decisões, estas funções são refletidas em valores relacionados a condições. Atividades também, geralmente não podem iniciar, a menos que certos produtos e agentes estiverem disponíveis, e certas condições estiverem satisfeitas. Esta visão é mostrada na Figura A.1.

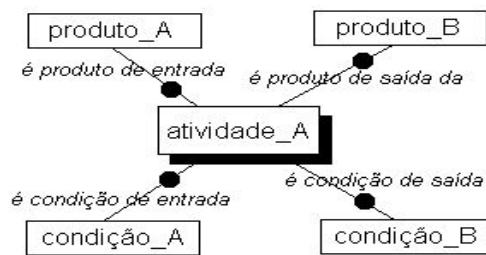


FIGURA A.1 - Representação básica de um elemento de processo.

FONTE: adaptada Christie (1993).

A Figura A.2 apresenta um simples exemplo de um processo de solicitação de mudança utilizando a linguagem ProNet, maiores informações ver (Christie 1993).

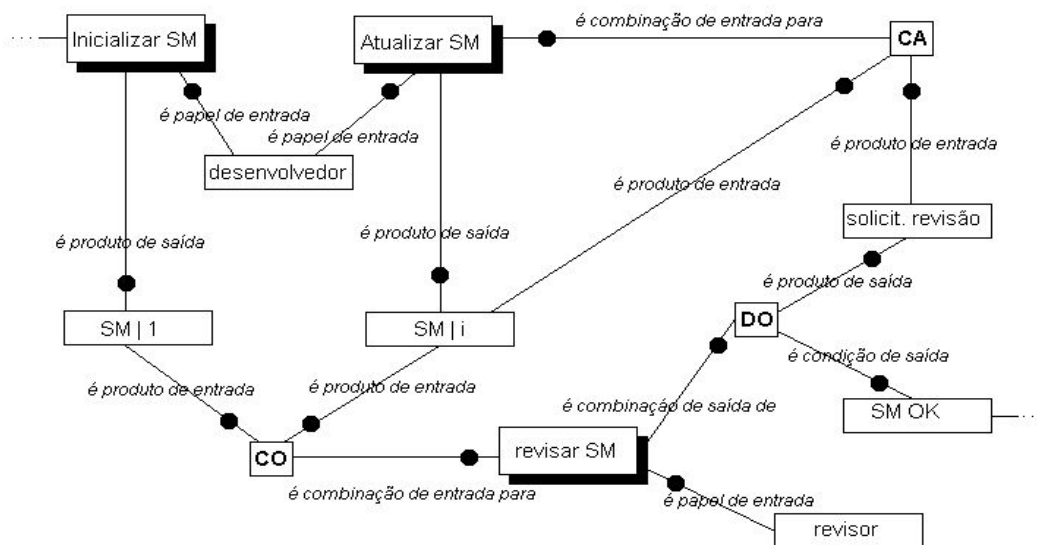


FIGURA A.2 - Um simples modelo de processo de solicitação de mudança.

FONTE: adaptada Christie (1993).

A.2.- A PML E3

A PML E³ ¹(Jaccheri et. al., 1998, 1999) tem suas raízes inspiradas em trabalhos na orientação a objetos. A linguagem oferece uma sintaxe formalmente definida que força a representação de modelos bem definidos. Uma representação gráfica é apresentada de modo a facilitar a representação e comunicação dos modelos de processo.

A PML E3 oferece um conjunto de classes e associações que foram concebidas para descrever as entidades básicas e associações envolvidas em processos de software. Estas classes e associações, por sua vez, podem ser customizadas pelo modelador por meio do uso de herança. A definição da linguagem expande-se em três níveis conceituais: nível de criação, nível de definição e nível de instância. Esta estrutura de três camadas, redefinem a distinção entre meta-classes, classes e instâncias que são apoiadas por algumas linguagens orientadas a objetos (OO).

A descrição da estrutura de três camadas da PML E3, descrita no parágrafo anterior, vem a seguir:

- **Nível de Criação:** Onde classes e associações são criadas por meio de especialização (herança).
- **Nível de Definição:** Onde os elementos criados no nível de criação são montados no intuito de se construir uma rede de classes conectadas por associações. Esta rede é atualmente a descrição de um Template².

¹ E³ significa Environment for Experimenting and Envolving software process

² Template: Captura aspectos-chave de regras definidas para a produção de software encontradas em um ou mais manuais de qualidade ou projeto, estes

- **Nível de Instância:** Onde classes e associações pertencentes a um Template definido no nível de definição, são instanciados em objetos e conexões (links), representando as entidades pertencentes a um modelo.

O *Nível de Criação*, inclui ambas as classes e associações pertencentes ao núcleo (kernel) ou definidas pelo modelador. O núcleo é imutável e constitui o nível mais alto da hierarquia de herança de todo template. O núcleo é organizado em três raízes na árvore hierárquica das classes da PML E³. As classes pertencentes ao núcleo são especializações da classe *E3Object*, as associações pertencentes ao núcleo, por sua vez, são especializações das associações *binária* ou *ternária*. Classes e associações criadas por usuários customizam o núcleo para um dado template. No *Nível de Definição*, as classes são conectadas por meio da utilização de associações, formando assim, uma rede de classes, que é a representação do template modelado.

A.2.1.- Classes e Associações pertencentes ao núcleo da PML E3

São apresentadas neste tópico a descrição de cada classe e associação pertencente ao núcleo da linguagem E³. Estas classes são organizadas em uma hierarquia de herança raizada a partir da classe *E3Object*.

A descrição das classes do núcleo da linguagem são descritas abaixo:

- **E3Object:** é a classe mais alto nível do nível hierárquico do núcleo da PML E3. As demais classes pertencentes ao núcleo, são especializações desta classe.

manuais, por sua vez definem um conjunto de características gerais que devem ser adotados em uma organização ou projeto. Um Template (Jaccheri, 1999), portanto, define um modelo que serve como um guia para uma classe de processos.

- **Task:** especializações da classe *Task*, definem características comuns para um conjunto de atividades de um processo de software. Cada instância da classe *Task*, denota uma atividade de processo que pode terminar com ou sem aprovação. Uma instância desta classe pode ser iniciada quando suas predecessoras estiverem completadas e seus recursos (entradas e papéis), estiverem prontos. A noção de aprovação oferecida pela linguagem, pode ser utilizada para modelar uma tarefa para verificação e validação, esta tarefa pode tanto aprovar o resultado de uma respectiva tarefa de produção, ou solicitar outra iteração.
- **Role:** especializações da classe *Role* definem responsabilidades e representam um conjunto de habilidades para uma pessoa. Instâncias da classe *Role* modelam recursos atuais, suas responsabilidades e habilidades. Um papel de projetista de classes, deve representar uma pessoa que passou por um program de treinamento sobre Orientação a Objetos. Isto pode ser modelado por uma classe *OODesiner*, especializada da classe *Role*.
- **Data:** especializações da classe *Data*, definem artefatos de um processo de software ou componentes de um produto como um código fonte ou um documento de revisão.
- **Tool:** especializações da classe *Tool* definem métodos de trabalho. Uma instância da classe *Tool* pode representar uma versão precisa de uma ferramenta automatizada ou um procedimento escrito.

As associações binárias pertencentes ao núcleo da linguagem E3 são as seguintes:

- **binary(E3Object' , E3Object'')**: é a classe de associação de nível mais alto na hierarquia de classes de associações binárias pertencentes ao núcleo da PML E3. Esta associação é super classe para cada associação definida entre duas classes E3Object.

- **aggregation(E3Object', E3Object'')**: representa composição. Definições da associação *aggregation* representam composição de classes e conexões (links) *aggregation* denotam composição de objetos. A associação binária *aggregation* fornecem uma das principais facilidades de abstração para representar como uma entidade pode ser refinada com a composição de outras entidades.
- **subtask(Task', Task'')**: herda as características da associação *aggregation* e denota composição entre subclasses *Task*. A associação *subtask* oferece uma das principais dimensões organizacionais de um modelo de processo. Decomposições de atividades é um modo bem conhecido de se estruturar divisão de trabalho em um processo de produção.
- **preorder(Task', Task'')**: declara regras de interações entre instâncias de tarefas. Se existe uma conexão *preorder(Task_01, Task_02)* significa que o objeto *Task_02* pode ser iniciada somente após o objeto *Task_01* ser completado com aprovação. Esta regra é similar a um relacionamento *finish-to-start* em gerenciamento de projetos.
- **feedback(Task', Task'')**: declara regras tanto para a instanciação e interação de instâncias de tarefas (classes *Task*). O conceito de *feedback*, foi introduzido na linguagem de modo a possibilitar a modelagem de repetições de atividades que podem ocorrer em um template. Uma conexão *feedback(Task_01, Task_02)* significa que se a instância *Task_01* for completada sem aprovação, uma instância *Task_02* deve ser iniciada, caso contrário, se a instância *Task_01* for completada com aprovação, uma instância *Task_02* não precisa ser necessariamente inicializada.
- **responsible(Task, Role)**: declara regras de interação entre recursos e atividades. Uma definição de associação *responsible(Task1, Role1)*,

informa que um conjunto de responsabilidades declaradas para uma classe *Role1* são necessárias para a execução de uma instância *Task1*.

- **output(Task, Data)** e **input(Task, Data)**: descrevem informações sobre documentos que são produzidos e consumidos por classes *Task*. Uma associação *output(Task1, Data1)* informa que uma instância de *Task1* produz como saída instâncias de *Data1*, e o mesmo se aplica analogamente para associações *input*.

São associações ternárias do núcleo da linguagem E3 as seguintes:

- **ternary(E3Object', E3Object'', E3Object''')**: é a classe de associação de nível mais alto na hierarquia de classes de associações ternárias pertencentes ao núcleo da PML E3. Esta associação é super classe para cada associação definida entre três classes E3Object.
- **output(Task, Data, Tool)**: Uma definição de associação *output(Task1, Data1, Tool1)* representa que uma instância *Task1* produz como saída instâncias *Data1* explorando instâncias *Tool1*.
- **input(Task, Data, Tool)**: Uma definição de associação *output(Task1, Data1, Tool1)* representa que uma instância *Task1* possui como entrada instâncias *Data1* explorando instâncias *Tool1*.

A.2.2.- Notação Gráfica

A notação gráfica das classes pertencentes ao núcleo da PML E3 são mostradas na Figura A.3.

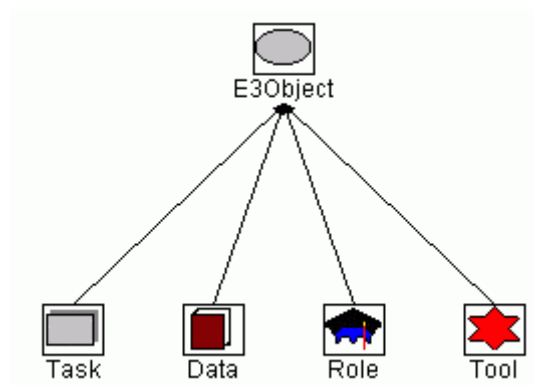


FIGURA A.3 - Hierarquia de classes pertencentes ao núcleo da PML E3 (Nível de Criação).

FONTE: Jaccheri et. al. (1998).

As notações gráficas das associações binárias e ternárias pertencente ao núcleo da PML E3 (Nível de criação) é mostrada na Figura A.4.

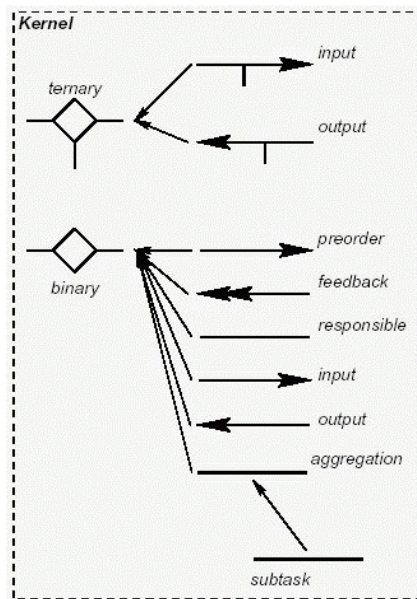


FIGURA A.4 - Associações do núcleo da PML E3.

FONTE: Jaccheri et. al. (1998).

A Figura A.5 apresenta um exemplo de uma rede de classes modelado com a PML E3 (Nível de Definição).

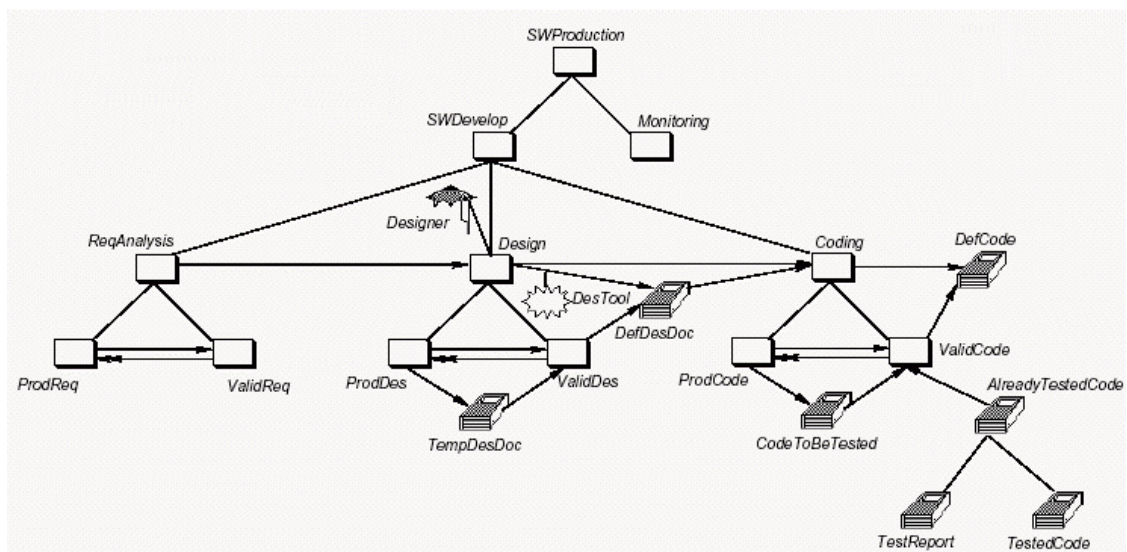


FIGURA A.5 - Rede de Classes (Nível de Definição) de um processo modelado com a PML E3.

FONTE: Jaccheri et. al. (1998).

A.3.- A PML PROMENADE

Um modelo para um processo de desenvolvimento de software, isto é, um modelo de processo de software (SPM Software Process Model) é uma descrição deste processo expressada em alguma linguagem de modelagem de processos (PML). O processo pode ser visto como a cooperação de muitas *tarefas* (ex. elicitação de requisitos, teste de componentes) que utilizam e desenvolvem alguns documentos (ex. plano de teste), com o auxílio de algumas ferramentas (ex. ferramentas CASE, compiladores) e utilizando alguns recursos (bases de dados, redes de computadores). Tarefas envolvem múltiplos agentes (pessoas, mídia hardware) que assumem papéis específicos (ex. programador, gerente) no qual se coordenam por meio de alguma mídia de comunicação (fax, e-mail). Todos estes conceitos já foram mencionados anteriormente, mas torna-se necessário uma pequena revisão para o entendimento do contexto da PML PROMENADE, portanto, a definição de um SPM deve representar todos os elementos mencionados, e, também, o modo com o qual o modelo deve ser executado “enacted”. Esta idéia nos deixa uma noção clara que um modelo de processo possui partes estáticas e dinâmicas. A

parte estática é a descrição dos elementos que fazem parte de um SPM. De outro lado, a parte dinâmica consiste na descrição do modo pelo qual o processo é executado. Uma descrição de ambas as partes não somente auxilia no entendimento do processo como também auxilia na construção de um sistema de apoio a automação do processo a um nível aceitável.

A abordagem PROMENADE (PROcess-oriented Modellization and ENAction of software DEvelopments), (Franch e Ribó, 1999a, 1999b), possibilita tanto a representação da parte estática de um SPM, quanto da parte dinâmica, onde a descrição da parte estática é fornecida por meio da utilização de diagramas da UML (Franch e Ribó, 1999a) e a descrição da parte dinâmica é feita por meio uma abordagem formulada para a PML PROMENADE onde combinam-se dois paradigmas complementares para o controle reativo e pró-ativo: relacionamentos de precedência entre tarefas, exceções e triggers; e na definição de mecanismos que melhoram a modularidade.

A.3.1.- A Parte Estática de um Processo de Software

A parte estática é construída sob o aspecto de três tipos de informação, representados por diagramas de classe UML, feitos com o auxílio de alguma ferramenta CASE. Primeiro, as informações individuais das classes, incluindo regras. Segundo, uma hierarquia de classes pelo qual são integrados todos os documentos por meio de herança (relacionamento de generalização da UML). Por último, outros relacionamentos de associação entre as classes, ambos agregação e definida pelo usuário.

A.3.1.1.- Classes e seus Membros

Na abordagem O.O. apresentada na PML PROMENADE, uma hierarquia de generalização de classes, é um modo natural de representar os principais conceitos envolvidos na modelagem de processos. Classes são caracterizadas por vários atributos e apoiadas por vários métodos. Valores válidos de atributos são sinalizados através de classes invariantes, enquanto métodos serão

especificados através de pré e pós condições. A Tabela A.3 apresenta estas classes.

TABELA A.3 - Classes pré-definidas da PML PROMENADE.

Classes	Descrição	Algumas Instâncias	Atributos	Alguns métodos
<i>Document</i>	Qualquer container de informação envolvido em um processo de desenvolvimento de software	Uma especificação; uma implementação de componente; um e-mail	Link para conteúdos, datas relevantes, versão, estado	Atualização de documento com ou sem criação de nova versão.
<i>Communication</i>	Qualquer documento utilizado para comunicação entre pessoas; pode ser armazenado ou não	Um fax, um e-mail, voz humana	Link para conteúdos, data de transmissão, estado	<i>Enviar e Ler</i>
<i>Task</i>	Qualquer ação executada durante o processo de desenvolvimento de software	Especificação; implementar componente; relatar erro	Pré-condição; estado; condição de sucesso; deadline	Mudanças de estado de tarefas
<i>Agent</i>	Qualquer entidade assumindo uma parte ativa no processo de desenvolvimento de software	Uma pessoa; uma estação de trabalho; um compilador	Perfil; localização; habilidades (para humanos)	-

(Continua)

TABELA A.3 – Conclusão.

<i>Tool</i>	Qualquer agente implementado por uma ferramenta de software	Um compilador; um navegador	Diretórios raiz para a ferramenta; localização do arquivo binário	-
<i>Resource</i>	Qualquer coisa que possa dar auxílio durante a execução de um processo de software.	Um tutorial on-line; um site da internet	Endereço; requisitos de plataforma	<i>Acessar</i>
<i>Role</i>	Qualquer parte a ser assumida durante o desenvolvimento de software	Programador; gerente	Tarefas para o qual o papel é responsável.	

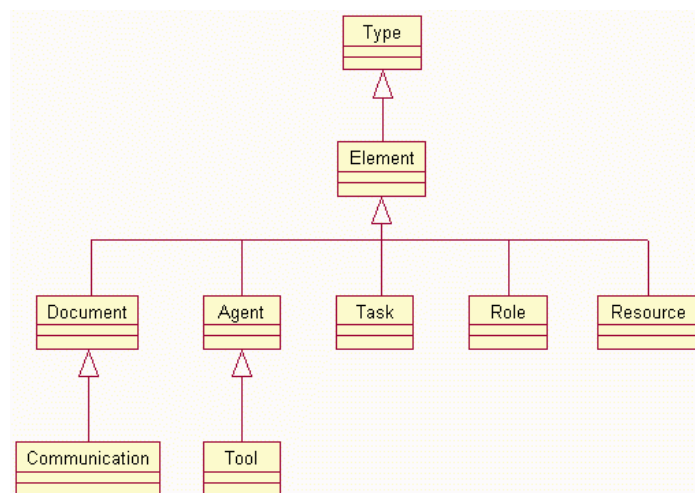


FIGURA A.6 - Hierarquia de generalização padrão.

A Figura A.7 mostra um diagrama de classes UML que representa algumas das associações existentes entre elementos de um modelo de processo.

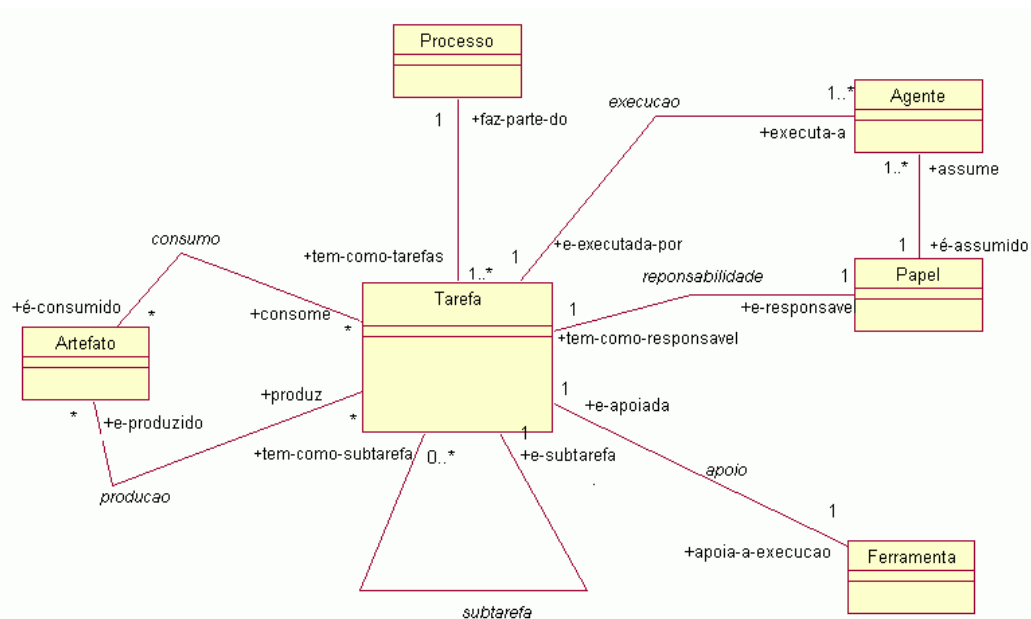


FIGURA A.7 - Relacionamentos entre elementos de processos.

FONTE: adaptada Franch e Ribó (1999a).

A.3.1.2.- A Parte Dinâmica de um Processo de Software: O Nível Intra-Modelo.

A parte dinâmica estabelece a ordem pelo qual as tarefas de um processo podem ser executadas “enacted”. A abordagem PROMENADE combina dois paradigmas diferentes de especificação de comportamento. De um lado estabelece-se relacionamentos de precedência para agendar apropriadamente a execução do modelo. De outro lado, permite-se a definição de gatilhos “triggers” e exceções “exceptions” que permitem a interrupção do processo a qualquer momento. Ambos os elementos são modelados pela PML usando diferentes mecanismos.

A.3.1.2.1.- Precedências

Precedências são o modo pelo qual são representadas a especificação de controle pró-ativo na abordagem PROMENADE, isto é, a especificação da

ordem de execução das atividades do processo. Muitas outras abordagens, implementam este tipo de controle de um modo imperativo, mas na abordagem PROMENADE, preferiu-se o estilo declarativo, porque ela atende melhor a natureza flexível da modelagem de processos. O layout das precedências é definida através de uma metaclassa UML que possui como principais atributos:

- **Tipos de precedência:** Em muitas situações diferentes, é um modo concreto para relacionar como as tarefas podem variar.
- **Precedências Fortes.** Existe uma forte precedência da tarefa s para a tarefa t se a finalização de s for necessária para a execução de t . Isto é um tipo usual de precedência quando consideramos o mesmo nível de abstração. Um exemplo típico pode ser encontrado na figura X, que apresenta uma forte precedência da tarefa *EspecificarComponente* para a tarefa *ImplementarComponente*.
- **Precedências fracas.** Existe uma fraca precedência da tarefa s para a tarefa t se s deve ser iniciada antes de t e s deve ser completada antes de t . Observe, entretanto, que ambas as condições fornecidas podem ser satisfeitas, s e t podem ser executadas ao mesmo tempo.
- **Precedências sincronizadas:** Distinguimos entre dois tipos diferentes. Existe um início de sincronismo na precedência da tarefa s para a tarefa t se s deve ser iniciada antes de t . Existe um final de sincronismo na precedência da tarefa s para a tarefa t se s deve ser completada antes de t . Observe que a precedência fraca está na soma do sincronismo de início e fim das precedências sincronizadas.
- **Acoplamento de Documentos.** Isto faz a associação de parâmetros envolvidos nas tarefas. Quando não chegam ambigüidades, documentos do mesmo tipo são acoplados por padrão.

<pre> precedence from SpecifyComponent to ImplementComponent: strong </pre>	<pre> precedence from EditImplementation to CompileImplementation: weak </pre>
---	--

FIGURA A.8 - Exemplos de Precências:

FONTE: adaptada Franch e Ribó (1999b).

A.3.1.2.2.- Agrupamentos

Agrupamentos são um modo alternativo para introduzir precedências entre tarefas. Esta construção trata da situação em que muitas tarefas devem ser consideradas ao todo: um agrupamento G de um conjunto de n tarefas estabelece aquele, uma vez que uma tarefa de G é executada; o processo deve terminar todas antes de começar qualquer outra tarefa (fora de G). Observe que o agrupamento não indica qualquer coisa sobre a ordem da execução das tarefas dentro de G , isto pode ser feito usando precedências. Deste modo, agrupar não obriga nem terminar uma tarefa antes de começar outra de G nem de outra maneira (por exemplo, as n tarefas podem ser ativadas simultaneamente se as precedências permitirem esta situação). A Figura A.9 mostra dois exemplos de agrupamentos.

<pre> grouping {FuncSpecifyComponent, NonFuncSpecifyComponent} grouping for all M: Component in ImportedBy(Self::spec_comp) {FuncSpecifyComponent(M)} </pre>

FIGURA A.9 - Exemplos de Agrupamentos.

FONTE: adaptada Franch e Ribó (1999b).

É fácil provar que os agrupamentos podem ser expressados nos termos das precedências. Entretanto, elas são mantidas na linguagem para finalidades do entendimento.

A.3.1.2.3.- Precedências Dinâmicas

Os relacionamentos de precedência são parte do modelo, e devem ser introduzidos no modelo em tempo da definição. Alguns deles, entretanto, podem somente ser determinados no tempo da execução “enaction”, quando o modelo é instanciado em um contexto particular. Considere por exemplo o seguinte relacionamento de precedência: a fim validar uma especificação funcional de um componente, requer-se validar cada operação individual desse componente. Está desobstruído que nós não sabemos adiantado quantas operações um componente terá. Lide com esta limitação, nós permitimos que a definição de relacionamentos dinâmicos da precedência. Uma precedência dinâmica está para um número desconhecido de relacionamentos "normais" da precedência; o número concreto será fixo durante a execução “enaction”, dependendo do valor das expressões dadas nos termos de um ou de mais atributos. Por exemplo, a Figura A.10 mostra o código de PROMENADE para a sentença precedente; o `spec_comp` é um atributo da tarefa (da classe `SpecComp`, isto é especificações dos componentes).

```
precedence dynamic
  for all op: Operation in Operations(Self::spec_comp)
    from SpecifyComponentOperation(Self::spec_comp, op)
    to ValidateSpecification(Self::spec_comp):
  strong
```

FIGURA A.10 - Um exemplo de precedência dinâmica.

FONTE: adaptada Franch e Ribó (1999b).

A.3.2.- A UML como Alternativa à Modelagem de Processos

Diversas características de metodologias orientadas a objetos: herança, agregação, reuso, relacionamentos podem ser muito úteis para modelagem de processos (Jaccheri et. al., 1998) (Georgakopoulos e Hornick, 1995) (Mühlen e Becker, 1999) (Schleicher et. al., 1998, 2000, 2001).

Alguns elementos podem ser encontrados na UML, onde, objetos que possuem semânticas específicas como, por exemplo, “Atores”, que são os papéis que as

peças exercem na execução de uma tarefa do processo. Outro importante conceito refere-se aos “Casos de Uso” (Use-cases), que podem essencialmente, ser utilizado para descrever as tarefas que compõem o processo (Georgakopoulos e Hornick, 1995) (Schleicher et. al., 1998, 2000, 2001).

A UML oferece 9 (nove) tipos de diagramas que podem ser utilizados para a modelagem de diferentes aspectos de um sistema sendo 4 (quatro) diagramas (diagrama de classes, diagrama de objetos, diagrama de componentes e diagrama de implantação) utilizados para a modelagem de aspectos estáticos e 5 (cinco) diagramas (diagrama de casos de uso, diagramas de seqüência, diagramas de colaborações, diagramas de gráficos de estado e diagramas de atividades), utilizados para a modelagem de aspectos dinâmicos (Jacobson et. al., 1999a).

Porém a UML é uma linguagem padrão, utilizada para modelagem de sistemas orientados a objetos, e, portanto não foi especificada para apoiar a modelagem de processos de software. No entanto a UML permite que seu meta-modelo seja estendido, permitindo expressar as diferentes nuances encontradas em diferentes domínios de aplicações (Jacobson et. al., 1999a).

Alguns estudos convergem para extensão do meta-modelo UML, para que o mesmo apóie a modelagem de processos. Criando novos elementos a partir de mecanismos de extensibilidade. Estes mecanismos são respectivamente: Estereótipos, Valores atribuídos e Restrições ver (Jacobson et. al., 1999a) (Schleicher et. al., 2001) (Franch e Ribó, 1999a).

Um estereótipo é uma meta classe UML virtual redefinida, ao qual, habilita o modelador a fazer a modelagem de elementos adicionais com semânticas definidas, padronizando a modelagem de um dado domínio de aplicações (SCH2000). Um "framework" utilizado para modelagem de processos é apresentada na Figura A.11.

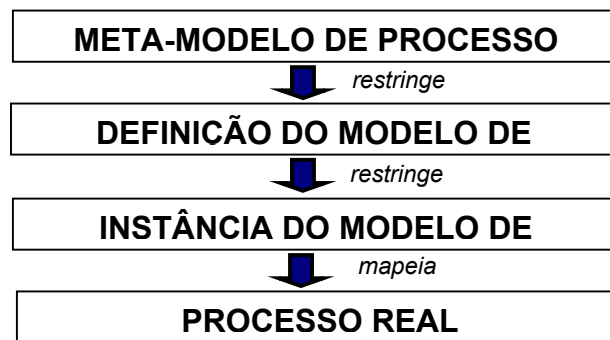


FIGURA A.11 - Framework para a modelagem de processos.

Schleicher (et. al., 1998, 2000, 2001), apresenta um conjunto de estereótipos definidos para modelagem dos elementos encontrados nos processos, dentre estes elementos encontram-se classes de objetos e associações aos quais são descritas na Tabela A.4, as regras para utilização das associações utilizadas em relacionamentos com os elementos são apresentadas na Tabela A.5:

TABELA A.4 - Estereótipos UML para a modelagem de processos.

Meta-Modelo UML	Estereótipo	Símbolo	Descrição
Classe	<<task>>		Meta classe tarefa
Classe	<<input>>		Define uma entrada para uma tarefa (ex. artefato)
Classe	<<output>>		Representa o produto de uma tarefa
Classe	<<realization>>		
Package	...		
Associação	<<cflow>>		Relacionameto tipo fluxo de controle para uma tarefa
Associação	<<fbback>>		Relacionamento tipo feedback para uma tarefa
Associação	<<dflow>>		Relacionamento tipo fluxo de dados, utilizado entre entradas e produtos de tarefa
Associação	<<may_contain>>		
Associação	<<may_have>>		Descrimina que uma tarefa pode conter um determinado artefato.
Associação	<<may_realize>>		

TABELA A.5 - Mapeamento elementos X associações.

De/Para	Task	Input	Output	Realization
Task	<<cfow>>, <<fback>>	<<may_have>>	<<may_have>>	
Input		<<dflow>>		
Output		<<dflow>>	<<dflow>>	
Realization	<<may_realize>>, <<may_contain>>			

A utilização de estereótipos torna possível a representação dos elementos pertinentes aos processos em objetos modelados em UML, como mencionado anteriormente, um modelo de processo deve representar a estrutura do processo bem como seu comportamento durante sua execução.

A modelagem de aspectos estáticos e dinâmicos de um processo, estrutura e comportamento, foi abordada por diversos autores. O Rational Unified Process (RUP) (Jacobson, 1999b) e Franch (Franch e Ribó, 1999a), apresentam a modelagem do comportamento dinâmico de um processo por meio de diagramas de atividades. Georgakopoulos (1995), apresenta a utilização de diagramas de casos de para representar a seqüência das tarefas que compõem um processo de workflow.

Schleicher (et. al., 1998, 2000, 2001) apresenta uma abordagem mais detalhada para definição de um modelo de processo utilizando a UML, a modelagem se dá por meio da construção de modelos, através da modelagem estrutural e comportamental de um modelo executável, onde são utilizadas os estereótipos apresentados acima em objetos presentes, em diagramas de casos de uso, objetos, classes, pacotes, colaboração e estados. A abordagem definida por schleicher é apresentada na Figura A.12.

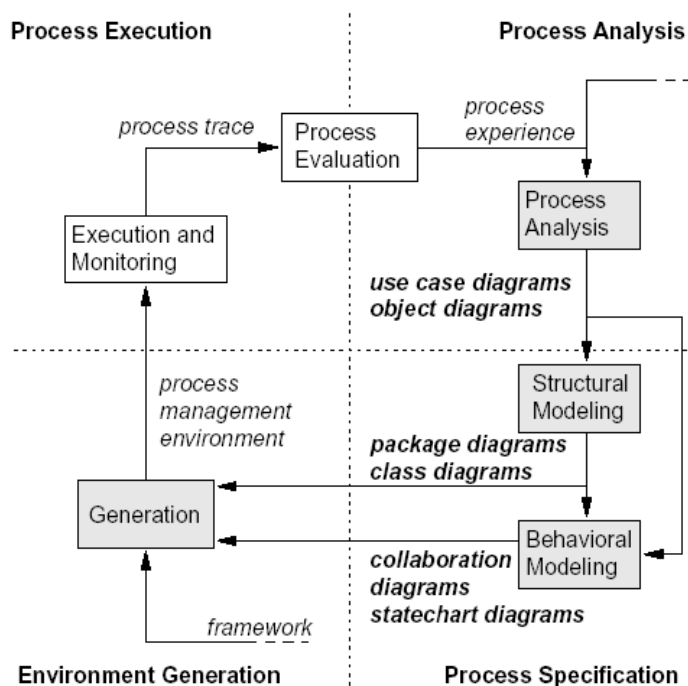


FIGURA A.12 - Abordagem para modelagem de processos apresentada por schleicher.

FONTE: Shleicher et. al. (2000).

A.4.- UPM – Modelo de Processo Unificado

O UPM - Modelo de Processo Unificado (OMG, 2000) é uma proposta inicial de submissão da OMG para um modelo usado na descrição de um concreto processo de desenvolvimento de software ou uma família relacionada de processos de desenvolvimento de software. A execução do processo está fora da extensão do UPM.

Diferente da maioria das PML o UPM aparece como uma proposta de unificação entre as diferentes metodologias propostas para modelagem de processos. Da mesma forma que a modelagem orientada o objetos despertou uma série de propostas, o que acabou por ocasionar a chamada guerra dos modelos, a modelagem de processos está passando pelo mesmo problema.

Organizações, pesquisadores, universidades criam suas metodologias e nomeiam os elementos dos processos com diferentes termos causando grande dificuldade para o desenvolvimento de produtos e da pesquisa na área de desenvolvimento de software.

O UPM utiliza uma aproximação orientada a objeto para modelar uma família relacionada de processos e usa a UML como notação. A Figura A.13 mostra as quatro arquiteturas de modelagem definidas pelo OMG.

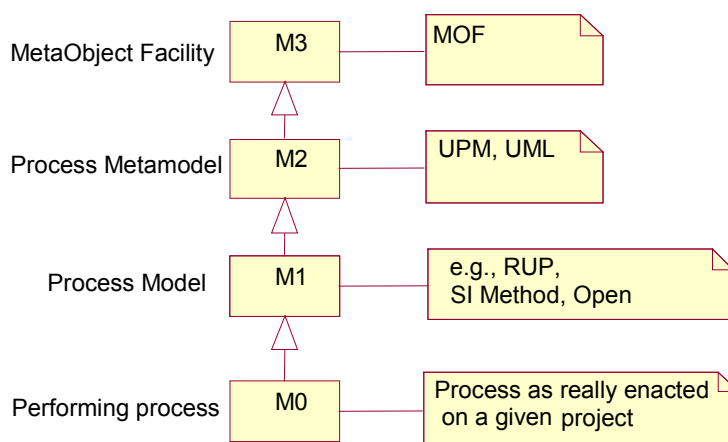


FIGURA A.13 - Níveis de Modelagem definidos pela OMG.

FONTE: OMG (2001).

O UPM encontra-se no nível M2 da arquitetura de quatro camadas e seu metamodelo é definido usando um subconjunto da UML de modo semelhante da UML com o MOF. Este subconjunto da UML corresponde às facilidades apoiadas pelo MOF. O metamodelo da UPM é largamente independente do metamodelo da UML, com a exceção do uso de Diagramas de Atividade.

Um processo executado é aquele, na produção no mundo real ou como o processo é ordenado, sendo nivelado no nível M0. A definição do processo correspondente está nivelado em M1. Por exemplo, o Processo Unificado Rational (Jacobson et. al. 1999b) ou o Método IBM SI estão definidos no nível M1. Ambos, processos genéricos, como RUP e uma específica customização deste processo usados por um determinado projeto, estão nivelado no M1. Nós

focalizamos aqui o metamodelo que representa a nivelção M2 e servindo como um template para o nível M1.

O UPM pode ser usado para definir todos os tipos de processos, incluindo os que estejam focalizados no uso específico da UML. Instâncias para orientar subclasses para descrever praticas com a UML podem ser criadas com ferramentas para um processo UML específico.

A.4.1.1.- Escopo do UPM

O UPM é um metamodelo para definir processos e seus componentes. Uma ferramenta baseada no UPM seria uma ferramenta para autoria e customização de processo. A atual execução de um processo, planejar e executar um projeto usando um processo descrito com o UPM, não está na escopo deste modelo.

A proposta inicial da OMG limita-se a definir o conjunto mínimo de processo que modela elementos necessário para descrever qualquer processo de desenvolvimento de software, sem adicionar modelos específicos ou condições para qualquer área específica ou disciplina, como gerenciamento de projeto ou análise.

A.4.1.2.- Elementos Principais

O metamodelo UPM é dividido em seis pacotes, chamados: Nomes , Elementos Básicos, Estrutura de Processo, Componentes de Processo, Orientação e Ciclo de Vida do Processo.

O pacote Elementos Básicos, detalhado na Figura A.14, define os elementos básicos dos quais o resto do modelo é derivado.

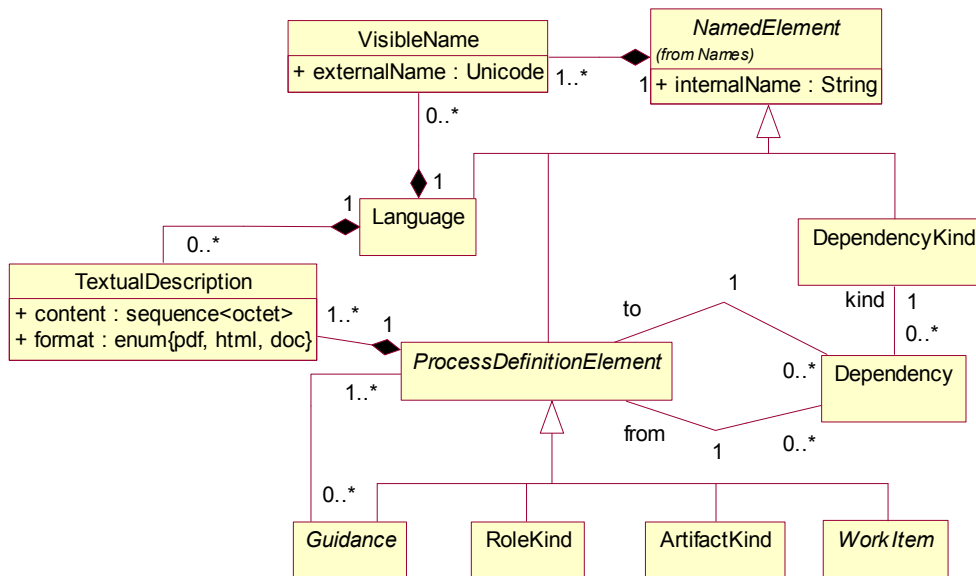


FIGURA A.14 - Pacote de Elementos básicos.

FONTE: OMG (2000).

Os principais elementos do pacote básico são:

- NamedElement, VisibleName, e Language:** A maioria dos elementos do metamodelo de processo são uma especialização de uma classe abstrata comum, chamada NamedElement, com um único atributo: internalName . Isto permite que elementos tenham um nome textual pelo qual podem se referir no modelo de processo.

Para permitir a expressão de um processo em qualquer idioma natural, um elemento nomeado pode ser associado a qualquer número de VisibleName que contém uma string de Unicode do nome do elemento em um idioma natural. Uma associação para o Idioma de classe especifica o idioma.

- ProcessDefinitionElement e TextualDescription:** A maioria dos elementos do metamodelo de processo também são uma especialização de uma classe abstrata comum, chamada ProcessDefinitionElement , introduzida para capturar atributos comuns como uma descrição textual. Em particular, todos os elementos de

processo têm um ou várias descrições em idiomas naturais. WorkDefinition, ArtifactKind, RoleKind, e Guidance (Orientações) são principais subdivisões de classe de ProcessDefinitionElement.

- **ArtifactKind e ArtifactName:** Um Artefato é qualquer coisa produzido, consumiu ou modificada por um processo. Pode ser um pedaço de informação, um documento, um modelo, código de fonte, e assim por diante. Um ArtifactKind descreve um tipo de artefato.
- **Guidance (Orientação):** um ou mais elementos de Guidance (orientação) são associados com um elemento de processo para prover ajuda. Guidance.
- **WorkItem, WorkDefinition, e WorkDefinitionName:** Um WorkItem é um ProcessDefinitionElement que descreve o trabalho executado através de papéis. É uma classe abstrata, com duas subdivisões de classe: WorkDefinition e Step. WorkItems podem ser usados em diagramas de atividade.

WorkDefinition é um ProcessDefinitionElement, uma subdivisão da classe WorkItem que descreve o trabalho executada por papéis. Suas subdivisões de classe principais são ActivityKind, como também Phase, Iteration, e Lifecycle (no Processo empacotam Lifecycle). Distintamente WorkItem e WorkDefinition podem ser recursivamente estruturados, e tem explicitamente entradas e saídas.

Um WorkDefinitionName é uma classe auxiliar cujas instâncias designam definições de trabalho ao criar definições de trabalho agregado. A introdução destas classes auxiliares permitem a um WorkDefinition a se referir com múltiplos nomes, semelhantemente ao ArtifactName para os Artefacts.

- **ActivityKind e Steps:** ActivityKind é a principal subclasse concreta de WorkDefinition. Descreve uma parte do trabalho executado por um papel: as tarefas, operações, e ações que são executadas por um papel ou com que o papel pode assistir. Um ActivityKind pode ser decomposto em elementos atômicos chamado Steps.

Steps são uma subdivisão de classe de WorkItem e, então, não podem ser decompostos. Steps são executados pelo mesmo papel que executa a atividade. Relationships

- **RoleKind:** Um RoleKind define um papel, possibilita a composto de um papel por uma pessoa, ou um grupo das pessoas, pode ser chamado para atuar em um processo. RoleKind define responsabilidades sobre os artefatos específicos, e define os papéis que executam e assistem as atividades específicas.





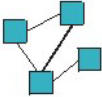
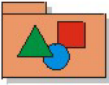
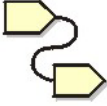


A.4.2.- Notação Gráfica

A intenção do UPM é oferecer uma notação gráfica para descrever processos de engenharia de software que são similares aos da UML. Porém, há pouco diagramas da UML que são prontamente possíveis, isso ocorre por causa das dificuldades de se fazer um perfil para UPM com a UML. Assim a proposta da OMG sobre o UPM não faz um perfil da UML para este. Espera-se que nas próximas submissões seja introduzidos esse mapeamento para mostrar a correspondência entre entidades de UPM e UML.

Idealmente uma notação gráfica para o UPM usaria diagramas de classe para descrever dependências entre elementos de processo, como uma (work-breakdown) estrutura de trabalhar ou estrutura de produto. Usaria diagrama de atividades para descrever a seqüência de atividades ou diagramas de colaboração para mostrar interação entre vários roles (papéis)

Os estereótipos propostos do UPM para uso em diagramas UML são demonstrados na Tabela A.6.

TABELA A.6 - Estereótipos propostos para gráficos.

UPM Element	Icon	Comments
<i>RoleKind</i>		
<i>ActivityKind</i>		
<i>ArtifactKind</i>	   	<i>Simple artifact,</i> <i>Document</i> <i>Model</i> <i>Aggregate of artifacts</i>
<i>WorkDefinition</i>		<i>Other than ActivityKind</i>
<i>ProcessComponent</i>		
<i>Process</i>		

FONTE: OMG (2000).

APÊNDICE B

CÓDIGO FONTE DO AGENTE MENSAGEIRO

Apresenta-se neste apêndice, o código fonte do agente mensageiro, desenvolvido para a plataforma JADE.

B.1.- Código Fonte do Agente Mensageiro

```
package ewebproject.processcoordinationsservice.processengine.agentservice;

/* bibliotecas externas */
import java.util.Vector;
import javax.mail.*;
import javax.mail.internet.*;
import jade.core.*;
import jade.core.behaviours.*;
import jade.domain.*;
import jade.content.lang.Codec;
import jade.content.*;
import jade.content.abs.*;
import jade.content.onto.*;
import jade.content.onto.basic.*;
import jade.content.lang.sl.*;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import jade.content.onto.basic.*;
import jade.util.leap.*;

/* bibliotecas internas */
import ewebproject.util.*;
import ewebproject.processcoordinationsservice.processengine.instancerepository.*;

/*
 * <p>Agente Mensageiro: Responsavel por notificar Agentes Humanos sobre ativacao de tarefas</p>
 * <p>Inicia seu trabalho a partir de uma mensagem recebida do Agente Coordenador</p>
 */
* @author Moacyr Gonçalves Cereja Junior
* @version 1.0
**/
public class MessengerAgent extends Agent
{
    /*
     * representa o comportamento interno
     * do agente mensageiro
     */
    * @author Moacyr Gonçalves Cereja Junior
    **/
    class NotifyHumanAgentsBehaviour extends SimpleBehaviour
    {
        // recupera configuracao de servidor SMTP
        private static String smtpHost = ServiceLocator.getInstance().getSMTPHost();

        /**
         * Construtor
         */
        public NotifyHumanAgentsBehaviour(Agent agent)
        {
            super(agent);
        }
    }
}
```

```

/**
 * Acao a ser executada quando o
 * agente coordenador solicitar a
 * notificacao de agentes humanos por
 * meio de uma mensagem ACL
 */
public void action()
{
    HumanAgents[] humanAgents = null;
    InternetAddress[] agentsToNotify = null;

    // Recebendo mensagem do Agente Coordenador
    MessageTemplate mt = MessageTemplate.and(
        MessageTemplate.MatchLanguage(codec.getName()),
        MessageTemplate.MatchOntology(ontology.getName()));
    ACLMessage msg = blockingReceive(mt);

    // extrair conteudo da mensagem
    try
    {
        // extraindo conteudo da mensagem
        ContentElement ce = getContentManager().extractContent(msg);

        if (ce instanceof Task)
        {
            Task task = (Task) ce;
            humanAgents = (HumanAgents[]) task.getHumanAgents().toArray();
            Vector addresses = new Vector();

            for (int i = 0; i < humanAgents.length; i++)
                addresses.addElement(new InternetAddress(humanAgents[i].getPerson().getEmail()));

            if (addresses.size() > 0)
            {
                agentsToNotify = new InternetAddress[addresses.size()];
                addresses.copyInto(agentsToNotify);
            }
        }
    }
    catch (CodecException ce) {
        Debug.log("Agente Mensageiro - Falha ao extrair conteudo da mensagem: " + ce.getMessage());
    }
    catch (OntologyException oe) {
        Debug.log("Agente Mensageiro - Falha ao extrair conteudo da mensagem: " + oe.getMessage());
    }
}

//notificar agentes humanos
try
{
    /* criar mensagem de e-mail */

    /* configuração do Host de SMTP */
    Properties properties = new Properties();
    properties.put("mail.smtp.host", smtpHost);

    /* estabelece uma sessao com servidora de e-mail */
    javax.mail.Session session = javax.mail.Session.getDefaultInstance(properties, null);
    session.setDebug(false);

    /* criar a mensagem */
    MimeMessage message = new MimeMessage(session);

    /* preencher assunto, endereco from e enderecos de envio */
    message.setSubject("There is a new task on your workspace");
    message.setFrom(new InternetAddress("ewebproject@sisis.com.br"));
    message.setRecipients(Message.RecipientType.TO, agentsToNotify);
}

```

```

/* atribui prioridade da mensagem para muito alta */
message.addHeader("X-Priority","1");

java.text.SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy.MMM.dd");

/* conteudo da mensagem */
StringBuffer() msgContent = new StringBuffer();
msgContent.append("Notification of Task activation \n\n");
msgContent.append("Task Name: " + Task.getName() + '\n');
msgContent.append("Activation Date: " + dateFormat.format(new Date()) + '\n');
for (int j = 0; j < humanAgents.length; j++)
    msgContent.append("Responsible: " + humanAgents[j].getPerson().getName() + '\n');

/* atribuir case a mensagem */
message.setText(msgContent.toString(), "iso-8859-1");

/* Enviar Mensagem */
Transport.send(message);
} catch (MessagingException me) {
    Debug.log("Falha ao enviar mensagem: " + me.getMessage());
} catch (AddressException ae) {
    Debug.log("Falha ao enviar mensagem: " + ae.getMessage());
}
}

} // fim metodo action

public boolean done()
{
    return true;
}

} //fim da classe NotifyHumanAgents

/* ATRIBUTOS DO AGENTE MENSAGEIRO */

/* linguagem para codificacao-decodificacao de objetos java em conteudo de mensagens ACL*/
private Codec codec = new SLCodec();

/* esquema-ontologia para o dominio de execucao de processos */
private Ontology ontology = ProcessEnactionOntology.getInstance();

/*
 * Configurando Agente
 */
protected void setup()
{
    // REGISTRO DA LINGUAGEM E ONTOLOGIA
    getContentManager().registerLanguage(codec);
    getContentManager().registerOntology(ontology);

    // ADICIONANDO COMPORTAMENTO AO AGENTE
    addBehaviour(new NotifyHumanAgentsBehaviour(this));
}

} // fim da classe MessengerAgent

```


APÊNDICE C

DESCRIÇÃO DOS CASOS DE USO E DIAGRAMAS DE SEQUÊNCIA DO AMBIENTE DE DEFINIÇÃO DE PROCESSOS DE SOFTWARE

Este apêndice apresenta, as descrições dos casos de uso e diagramas de seqüência apresentados no Capítulo 5.

C.1.- Descrição dos Casos de Uso

Segue abaixo a descrição de cada caso de uso pertinente ao Ambiente de Definição de Processos.

C.1.1.- Caso de Uso: Manter Modelos de Processo

Identificador do Caso de Uso	UC-PDE-01
Nome do Caso de Uso	Manter Modelos de Processos
Atores	Modelador de Processos
Prioridade	Muito Alta
Pré Condição	Um usuário de perfil administrador válido deve ter sido autenticado pelo sistema
Requisitos Especiais	Não há
<i>Primeiro Cenário – Adicionar Modelo de Processo</i>	
Fluxo de Eventos:	
Fluxo Principal 1.- Este caso de uso inicia-se quando o modelador de processos seleciona a opção “New Process Model”; 2.- O sistema apresentará um formulário com informações referentes ao processo: Nome, Descrição, Objetivo, etc; 3.- O modelador de processos entra com os dados no formulário; 4.- O modelador de processos seleciona Insert; 5.- O sistema irá verificar e armazenar as informações.	
Fluxos Alternativos No passo 5, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário	
Pos Condição: O sistema apresentara uma mensagem de sucesso da operação.	
<i>Segundo Cenário – Editar Modelo de Processo</i>	
Fluxo de Eventos:	
Fluxo Principal 1.- Este caso de uso inicia-se quando o modelador de processos seleciona um processo na arvore de processos no lado esquerdo da Interface principal; 2.- O sistema busca as informações na base de dados e apresenta uma tela	

<p>informando o nome do modelo e listando as tarefas que compõem o modelo; 3.- O modelador de processos seleciona "Edit Process Model"; 4.- O sistema apresenta um formulário com as informações atuais do modelo; 5.- O modelador de processos entra com as modificações; 6.- O modelador de processos seleciona "Update"; 7.- O sistema verifica e armazena as novas Informações;</p>
<p>Fluxos Alternativos No passo 7, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro.</p>
<p>Pós Condição: O sistema apresentara uma mensagem de sucesso da operação.</p>
<p>Terceiro Cenário – Excluir Modelo de Processo</p>
<p>Fluxo de Eventos:</p>
<p>Fluxo Principal 1.- Este caso de uso inicia-se quando o modelador de processos seleciona um processo na árvore de processos no lado esquerdo da Interface principal; 2.- O sistema busca as informações na base de dados e apresenta uma tela informando o nome do modelo e listando as tarefas que compõem o modelo; 3.- O modelador de processos seleciona "Delete Process Model"; 4.- O sistema apresenta uma mensagem para confirmação da exclusão; 5.- O modelador de processos seleciona Confirmar 6.- O sistema efetua a operação na base de dados;</p>
<p>Fluxos Alternativos No passo 4, se o modelador de processos selecionar "Não Excluir", o sistema apresenta a interface descrita no passo 4. No passo 6, se ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao cliente</p>
<p>Pos Condição: O sistema apresentara uma mensagem de sucesso da operação.</p>

C.1.2.- Caso de Uso: Manter Tarefa

Identificador do Caso de Uso	UC-PDE-02
Nome do Caso de Uso	Manter Tarefa
Atores	Modelador de Processos
Prioridade	Muito Alta
Pré Condição	O cenário Editar Modelo de Processo do caso de uso UC-PDE-01 deve ter sido executado previamente
Requisitos Especiais	Não há
Primeiro Cenário – Adicionar Modelo de Tarefa	
Fluxo de Eventos:	
Fluxo Principal 1.- Este caso de uso inicia-se quando o modelador de processos seleciona a opção "New Template Task" na tela principal de propriedades do modelo de processo; 2.- O sistema apresentará um formulário com informações referentes a tarefa; 3.- O modelador de processos entra com os dados no formulário; 4.- O modelador de processos seleciona Insert; 5.- O sistema irá verificar e armazenar as informações;	
Fluxos Alternativos No passo 5, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao	

usuário
Pós Condição: O sistema passará para o próximo passo da configuração da tarefa.
Segundo Cenário – Editar Modelo de Tarefa
Fluxo de Eventos:
Fluxo Principal 1.- Este caso de uso inicia-se quando o modelador de processos seleciona a opção “Edit” de uma tarefa listada na tela principal de propriedades do processo ou seleciona uma tarefa na árvore de processos; 2.- O sistema busca as informações na base de dados e apresenta uma tela informando o nome do modelo e listando as tarefas que compõem o modelo; 3.- O modelador de processos seleciona “Edit Process Model”; 4.- O sistema apresenta um formulário com as informações atuais do modelo; 5.- O modelador de processos entra com as modificações; 6.- O modelador de processos seleciona “Update”; 7.- O sistema verifica e armazena as novas Informações;
Fluxos Alternativos No passo 7, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro.
Pos Condição: O sistema apresentara uma mensagem de sucesso da operação.
Terceiro Cenário – Excluir Modelo de Tarefa
Fluxo de Eventos:
Fluxo Principal 1.- Este caso de uso inicia-se quando o modelador de processos seleciona um processo na árvore de processos no lado esquerdo da Interface principal; 2.- O sistema busca as informações na base de dados e apresenta uma tela de propriedades informando o nome do modelo e listando as tarefas que compõem o modelo; 3.-O modelador de processos seleciona as tarefas que deseja excluir do modelo, clicando nas caixas de seleção; 4.- O modelador de processos seleciona “Delete Template Task”; 5.- O sistema efetua a operação na base de dados;
Fluxos Alternativos No passo 5, se ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao cliente
Pós Condição: O sistema apresentara uma mensagem de sucesso da operação.

C.1.3.- Caso de Uso: Definir Papéis responsáveis

Identificador do Caso de Uso	UC-PDE-03
Nome do Caso de Uso	Definir Papéis Responsáveis
Atores	Modelador de Processos
Prioridade	Muito Alta
Pré Condição	O cenário Editar Modelo de Tarefa do caso de uso UC-PDE-02 deve ter sido executado previamente
Requisitos Especiais	Não há

Primeiro Cenário – Definir Papéis Responsáveis	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o modelador de processos seleciona a aba “Relationships” na tela de propriedades do modelo de tarefa; 2.- O sistema apresentará um formulário com duas listagens: uma listando os papéis disponíveis e outra listando os papéis responsáveis pela execução da tarefa; 3.- O modelador de processos seleciona na primeira lista os papéis que devem ser responsáveis pela tarefa e transfere para a segunda lista clicando no botão “>>”; 4.- O modelador de processos seleciona Update; 5.- O sistema irá verificar e armazenar as informações; 	
Fluxos Alternativos	
No passo 5, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário	
Pós Condição:	
O sistema passará para o próximo passo da configuração da tarefa.	

C.1.4.- Caso de Uso: Definir Recursos

Identificador do Caso de Uso	UC-PDE-04
Nome do Caso de Uso	Definir Recursos
Atores	Modelador de Processos
Prioridade	Moderada
Pré Condição	O cenário Editar Modelo de Tarefa do caso de uso UC-PDE-02 deve ter sido executado previamente
Requisitos Especiais	Não há
Primeiro Cenário – Definir Recursos	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o modelador de processos seleciona a aba “Resources” na tela de propriedades do modelo de tarefa; 2.- O sistema apresentará um formulário com caixas de listagens oferecendo como opção as opções de recursos e tipos disponíveis; 3.- O modelador de processos seleciona as opções desejadas e adiciona o recurso à tarefa clicando no botão “Add”; 4.- O sistema atualiza a lista de recursos utilizados e apresenta ao modelador de processos. 5.- O modelador de processos seleciona Update; 6.- O sistema irá verificar e armazenar as informações; 	
Fluxos Alternativos	
No passo 3 e 6, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário	
Pós Condição:	
O sistema passará para o próximo passo da configuração da tarefa.	

C.1.5.- Caso de Uso: Definir Tipos de Artefato de Entrada

Identificador do Caso de Uso	UC-PDE-05
Nome do Caso de Uso	Definir Tipos de Artefato de Entrada
Atores	Modelador de Processos
Prioridade	Moderada

Pré Condição	O cenário Editar Modelo de Tarefa do caso de uso UC-PDE-02 deve ter sido executado previamente
Requisitos Especiais	Não há
Primeiro Cenário – Definir Tipos de Artefato de Entrada	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o modelador de processos seleciona a aba “Input” na tela de propriedades do modelo de tarefa; 2.- O sistema apresentará um formulário com duas listagens: uma listando os tipos de artefatos disponíveis e outra listando os tipos de artefatos consumidos durante a execução da tarefa; 3.- O modelador de processos seleciona na primeira lista os tipos de artefatos que devem ser utilizados pela tarefa e transfere para a segunda lista clicando no botão “>>”; 4.- O modelador de processos seleciona Update; 5.- O sistema irá verificar e armazenar as informações; 	
Fluxos Alternativos	
No passo 5, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário	
Pós Condição:	
O sistema passará para o próximo passo da configuração da tarefa.	

C.1.6.- Caso de Uso: Definir Tipos de Artefato de Saída

Identificador do Caso de Uso	UC-PDE-06
Nome do Caso de Uso	Definir Tipos de Artefato de Saída
Atores	Modelador de Processos
Prioridade	Moderada
Pré Condição	O cenário Editar Modelo de Tarefa do caso de uso UC-PDE-02 deve ter sido executado previamente
Requisitos Especiais	Não há
Primeiro Cenário – Definir Tipos de Artefato de Entrada	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o modelador de processos seleciona a aba “Output” na tela de propriedades do modelo de tarefa; 2.- O sistema apresentará um formulário com duas listagens: uma listando os tipos de artefatos disponíveis e outra listando os tipos de artefatos produzidos durante a execução da tarefa; 3.- O modelador de processos seleciona na primeira lista os tipos de artefatos que devem ser produzidos pela tarefa e transfere para a segunda lista clicando no botão “>>”; 4.- O modelador de processos seleciona Update; 5.- O sistema irá verificar e armazenar as informações; 	
Fluxos Alternativos	
No passo 5, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário	
Pós Condição:	
O sistema passará para o próximo passo da configuração da tarefa.	

C.1.7.- Caso de Uso: Definir Relacionamentos entre Tarefas

Identificador do Caso de Uso	UC-PDE-07
Nome do Caso de Uso	Definir Tipos de Artefato de Saída
Atores	Modelador de Processos
Prioridade	Moderada
Pré Condição	O cenário Editar Modelo de Tarefa do caso de uso UC-PDE-02 deve ter sido executado previamente
Requisitos Especiais	Não há
<i>Primeiro Cenário – Definir Relacionamentos Tipo Sub-Tarefas</i>	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o modelador de processos seleciona a aba “Sub-Tasks” na tela de propriedades do modelo de tarefa; 2.- O sistema apresentará um formulário com duas listagens: uma listando as tarefas definidos para o processo sendo editado e outra listando as tarefas que são sub-tarefas da tarefa sendo editada; 3.- O modelador de processos seleciona na primeira lista tarefas que devem ser sub-tarefas e transfere para a segunda lista clicando no botão “>>”; 4.- O modelador de processos seleciona Update; 5.- O sistema irá verificar e armazenar as informações; 	
Fluxos Alternativos	
No passo 5, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário	
Pós Condição:	
O sistema passará para o próximo passo da configuração da tarefa.	
<i>Segundo Cenário – Definir Relacionamentos Tipo Predecessora</i>	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o modelador de processos seleciona a aba “Relationships” na tela de propriedades do modelo de tarefa; 2.- O sistema apresentará um formulário com caixas de listagens oferecendo como opção as opções de relacionamento e tipos disponíveis; 3.- O modelador de processos seleciona as opções desejadas e adiciona o relacionamento à tarefa clicando no botão “Add”; 4.- O sistema atualiza a lista de relacionamentos e apresenta ao modelador de processos. 5.- O modelador de processos seleciona Update; 6.- O sistema irá verificar e armazenar as informações; 	
Fluxos Alternativos	
No passo 3 e 6, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário	
Pós Condição:	
O sistema passará para o próximo passo da configuração da tarefa.	

C.2.- Modelagem da Interação entre os Objetos

Apresenta-se, nesta seção, os aspectos referentes a troca de mensagens entre os objetos necessários à realização de cada caso de uso definido na seção 5.2. A modelagem da interação dos objetos se dá por meio da utilização de diagramas de seqüência UML, definidos para cada cenário proposto ao caso de uso em questão.

C.2.1.- A Seqüência: Caso de Uso Manter Modelos de Processo

Seqüência de mensagens dos objetos que colaboram para realização do primeiro cenário Inserir Modelos de Processos mostrados na Figura C.1:

- 1) O Modelador de Processos entra com os dados do processo e envia as solicitações ao ambiente;
- 2) O ambiente aciona o Componente ProcessarModelosDeProcessos para Processar a operação de inserção no banco de dados;
- 3) O Componente cria uma conexão com banco de dados em modo transacional, cria um Objeto ModeloDeProcessos, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado;
- 4) O componente solicita ao objeto ModeloDeProcessos para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeProcessos;
- 5) O objeto ProcessarModelosDeProcessos, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário.

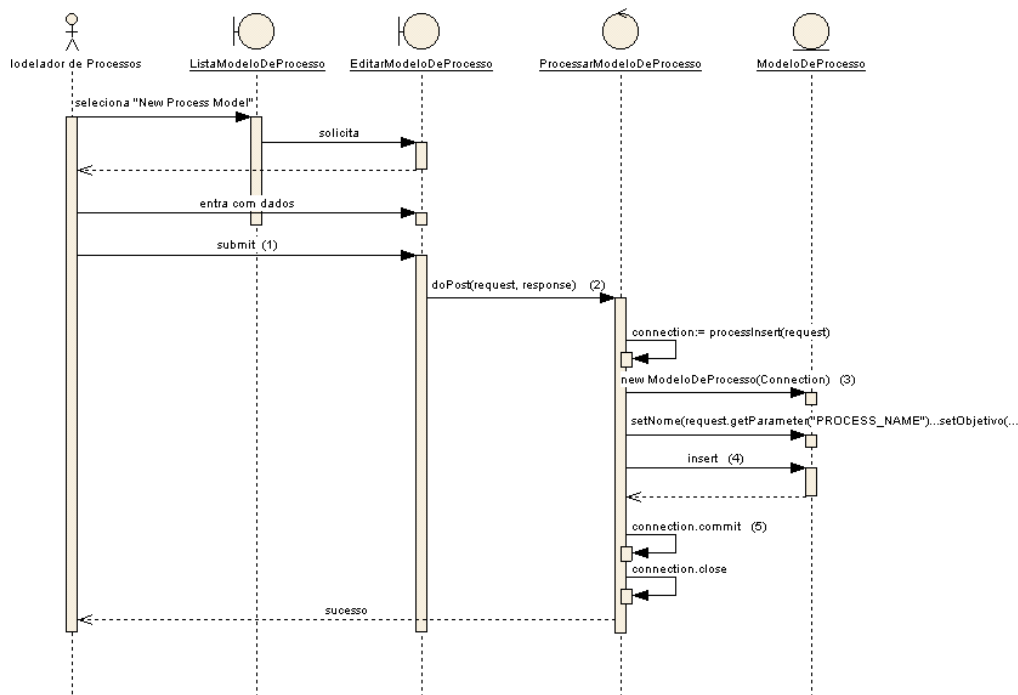


FIGURA C.1 - Diagrama de seqüência: inserir um novo modelo de processo.

Seqüência de mensagens dos objetos que colaboram para realização do primeiro cenário Editar Modelos de Processos mostrados na Figura C.2:

- 1) O Modelador de Processos solicita a edição de um ModeloDeProcesso, a Interface solicita ao objeto ModeloDeProcesso que recupere seus dados e imprime os dados no formulário;
- 2) O Modelador de Processos entra com os dados do processo e envia as solicitações ao ambiente;
- 3) O ambiente aciona o Componente ProcessarModelosDeProcessos para Processar a operação de atualização no banco de dados;
- 4) O Componente cria uma conexão com banco de dados em modo transacional, cria um Objeto ModeloDeProcessos, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado;

- 5) O componente solicita ao objeto ModeloDeProcessos para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeProcesso;
- 6) O objeto ProcessarModelosDeProcesso, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário

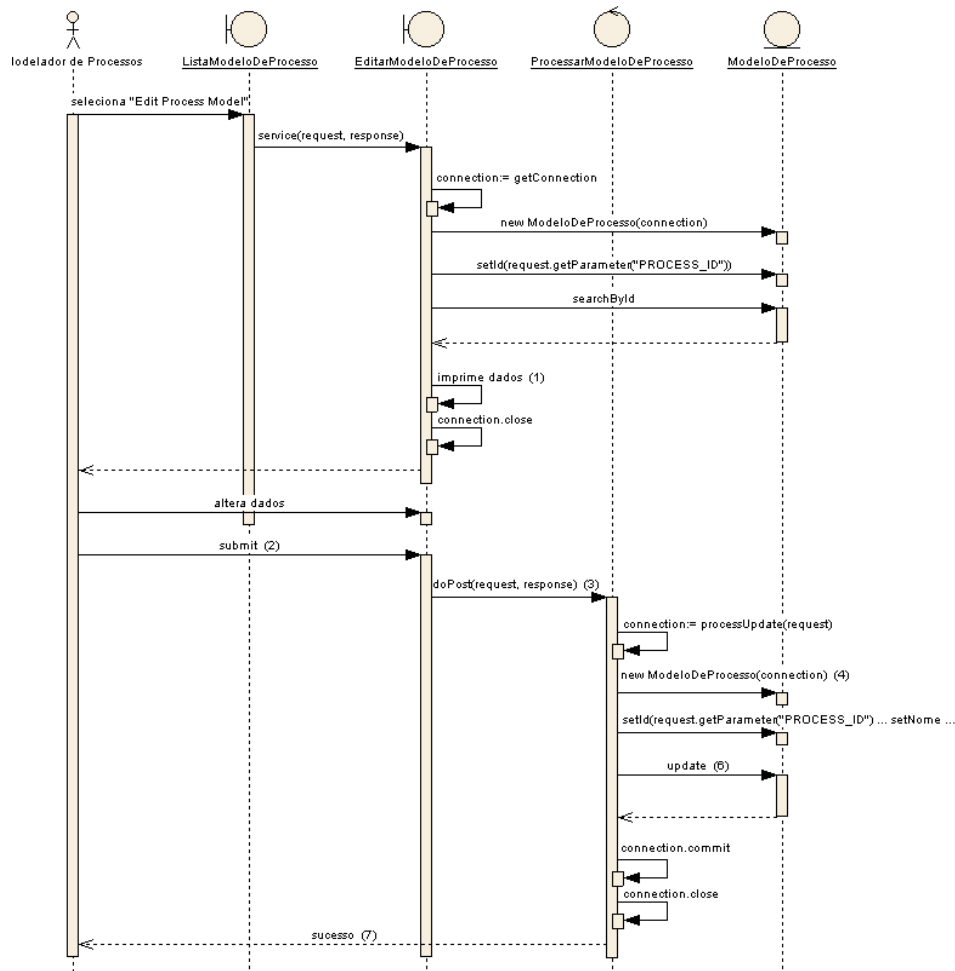


FIGURA C.2 - Diagrama de seqüência para editar um modelo de processo.

Seqüência de mensagens dos objetos que colaboram para realização do primeiro cenário Editar Modelos de Processos mostrados na Figura C.3:

- 1) O Modelador de Processos solicita a exclusão de um ModeloDeProcesso na interface principal;

- 2) A Interface principal solicita uma interface para confirmação de exclusão e apresenta ao Modelador de Processos, o usuário clica em OK, confirmando a operação;
- 3) O ambiente aciona o Componente ProcessarModelosDeProcessos para Processar a operação de exclusão no banco de dados;
- 4) O Componente cria uma conexão com banco de dados em modo transacional, cria um Objeto ModeloDeProcessos, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado;
- 5) O componente solicita ao objeto ModeloDeProcessos para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeProcessos;
- 6) O objeto ProcessarModelosDeProcessos, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário.

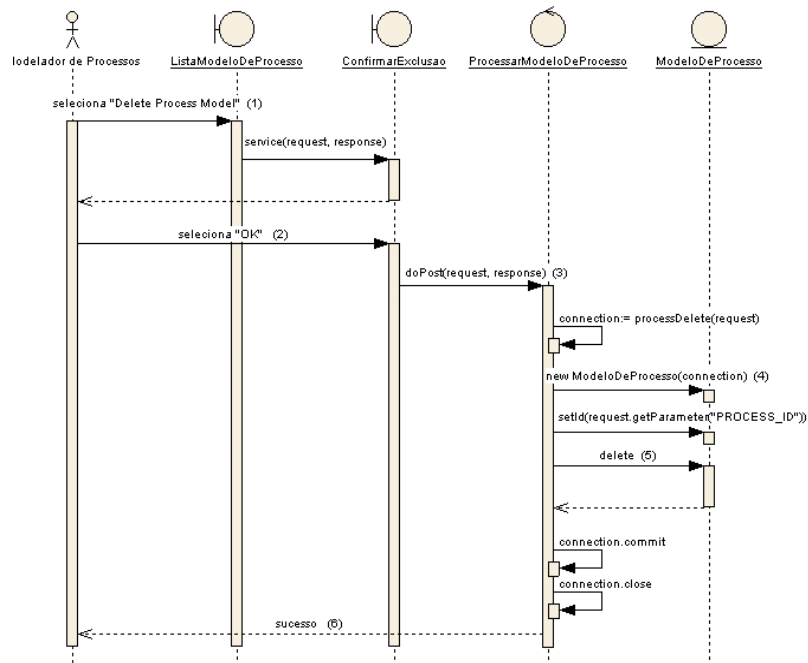


FIGURA C.3 - Diagrama de seqüência para excluir um modelo de processo.

C.2.2.- A Seqüência: Caso de Uso Manter Modelos de Tarefa

Seqüência de mensagens dos objetos que colaboram para realização do primeiro cenário Inserir Modelos de Processos mostrados na Figura C.4:

- 1) O Modelador de Processos entra com os dados da tarefa e envia as solicitações ao ambiente;
- 2) O ambiente aciona o Componente ProcessarModelosDeTarefa para Processar a operação de inserção no banco de dados;
- 3) O Componente cria uma conexão com banco de dados em modo transacional, cria um Objeto ModeloDeProcessos, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado, repete a operação com o objeto ModeloDeTarefa e atribui a Tarefa;
- 4) O componente solicita ao objeto ModeloDeTarefa para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeTarefa;
- 5) O objeto ProcessarModelosDeTarefa, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário.

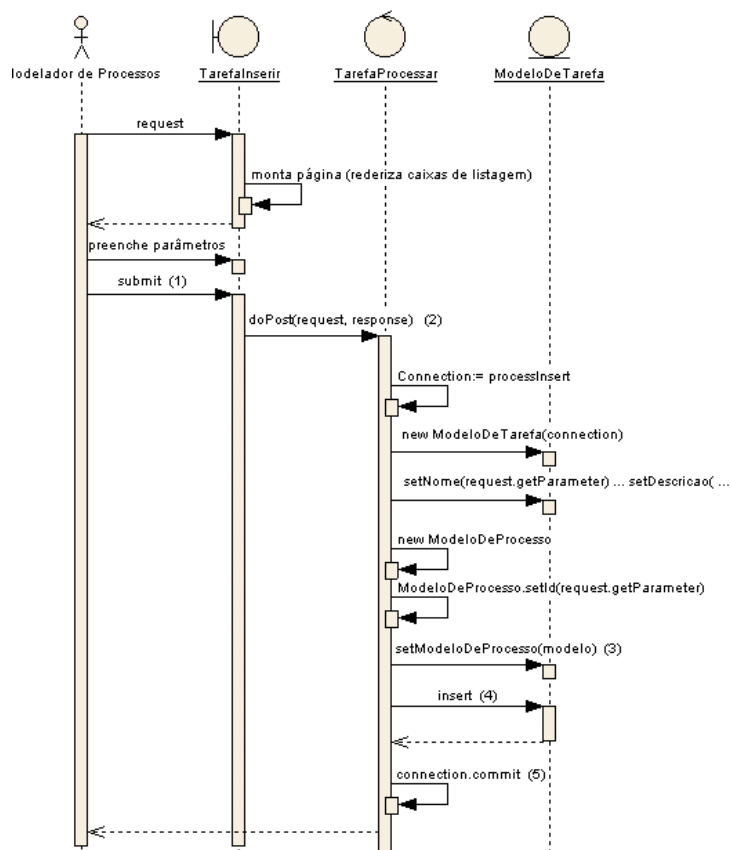


FIGURA C.4 - Diagrama de seqüência para inserir um modelo de tarefa.

Seqüência de mensagens dos objetos que colaboram para realização do primeiro cenário Editar Modelos de Tarefa mostrados na Figura C.5:

- 1) O Modelador de Processos solicita a edição de um ModeloDeTarefa, a Interface solicita ao objeto ModeloDeTarefa que recupere seus dados e imprime os dados no formulário;
- 2) O Modelador de Processos entra com os dados da tarefa e envia as solicitações ao ambiente;
- 3) O ambiente aciona o Componente ProcessarModelosDeTarefa para Processar a operação de atualização no banco de dados;

- 4) O Componente cria uma conexão com banco de dados em modo transacional, cria um Objeto ModeloDeTarefa, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado;
- 5) O componente solicita ao objeto ModeloDeTarefa para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeTarefa;
- 6) O objeto ProcessarModelosDeTarefa, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário

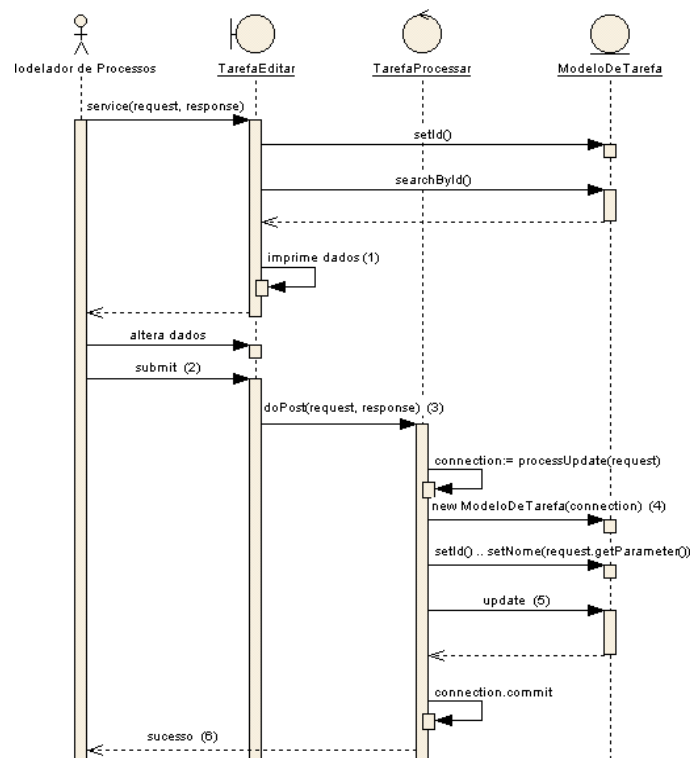


FIGURA C.5 - Diagrama de seqüência para editar um modelo de tarefa.

Seqüência de mensagens dos objetos que colaboram para realização do primeiro cenário Editar Modelos de Tarefa mostrados na Figura C.6:

- 1) O Modelador de Processos aciona a Interface ListarModelosDeProcesso que ao objeto ModeloDeProcesso que

recupere os objetos ModeloDeTarefa relacionados e imprime as informações;

- 2) O Modelador de Processos seleciona as tarefas que deseja excluir e clica em “Delete Task”;
- 3) O ambiente aciona o Componente ProcessarModelosDeTarefa para Processar a operação de exclusão no banco de dados;
- 4) O Componente cria uma conexão com banco de dados em modo transacional, cria um Objeto ModeloDeTarefa, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado;
- 5) O componente solicita ao objeto ModeloDeTarefa para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeTarefa;
- 6) O objeto ProcessarModelosDeTarefa, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário.

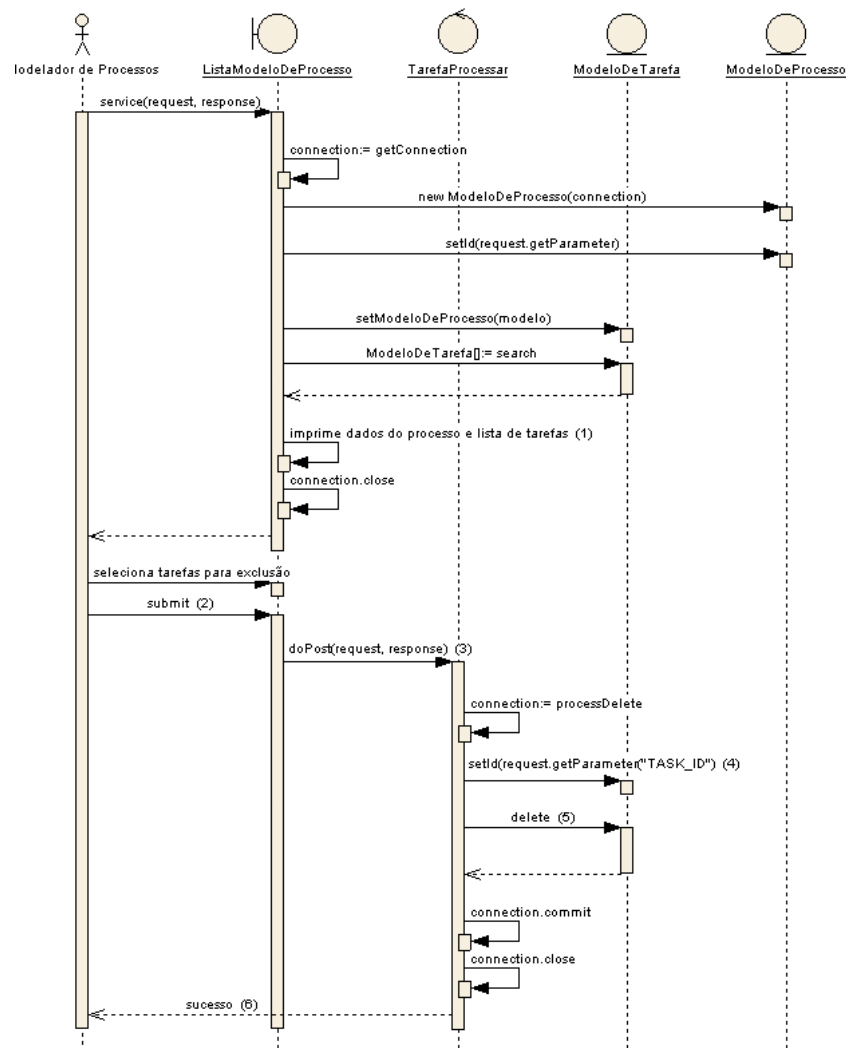


FIGURA C.6 - Diagrama de seqüência para excluir um modelo de tarefa.

C.2.3.- A Seqüência: Caso de Uso Definir Responsável

Seqüência de mensagens dos objetos que colaboram para realização do caso de uso Definir Responsável mostrados na Figura C.7:

- 1) Após selecionar a opção “Responsible” na Interface de Edição de ModeloDeTarefa, é acionada a interface de Edição de papéis responsáveis, a interface aciona o objeto papel que carrega os recupera os dados da base de dados, retornando o resultado que é carregado na tela;

- 2) O Modelador de Processos seleciona os papéis disponíveis na caixa de listagem e relaciona com o ModeloDeTarefa, submetendo as informações em seguida;
- 3) O ambiente aciona o Componente ProcessarModelosDeTarefa para Processar a operação de atualização no banco de dados;
- 4) O Componente cria uma conexão com banco de dados em modo transacional, cria um Objeto papel, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado e relaciona o objeto Papel com o ModeloDeTarefa que está sendo editado;
- 5) O componente solicita ao objeto ModeloDeTarefa para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeTarefa;
- 6) O objeto ProcessarModelosDeTarefa, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário.

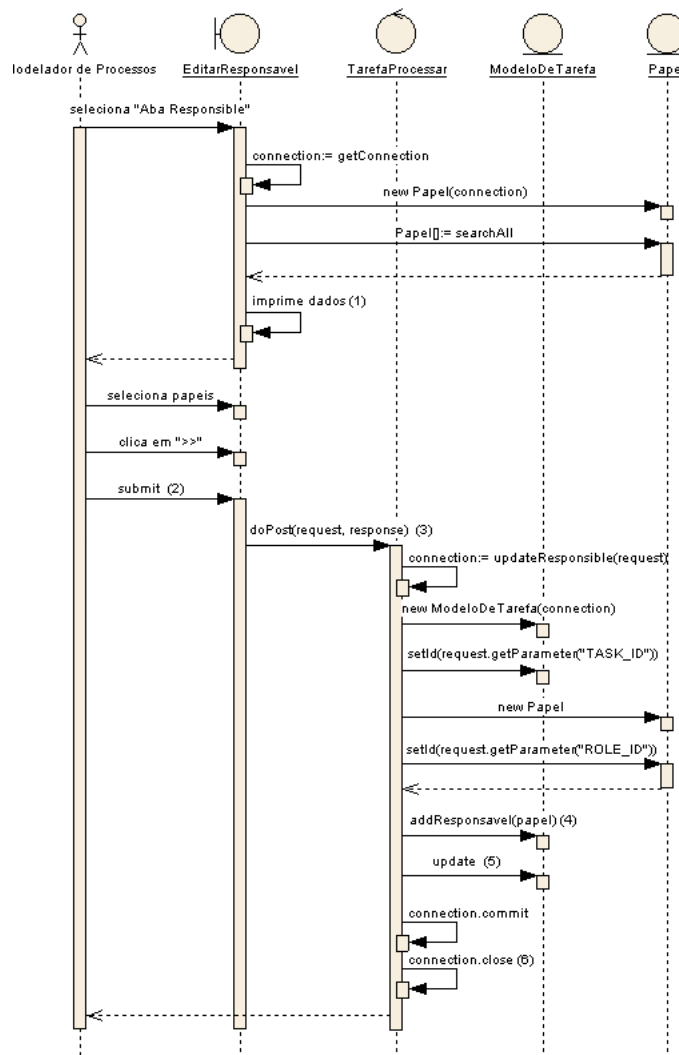


FIGURA C.7 - Diagrama de seqüência caso de uso Definir Responsáveis.

C.2.4.- A Seqüência: Caso de Uso Definir Recursos

Seqüência de mensagens dos objetos que colaboram para realização do caso de uso Definir Responsável mostrados na Figura C.8:

- 1) Após selecionar a opção "Resources" na Interface de Edição de ModeloDeTarefa, é acionada a interface de Edição de recursos, a interface aciona o objeto papel que carrega os recupera os dados da base de dados, retornando o resultado que é carregado na tela;

- 2) O Modelador de Processos seleciona os papéis disponíveis na caixa de listagem e relaciona com o ModeloDeTarefa clicando em “Add”, a interface atualiza a lista de recursos relacionados com a tarefa;
- 3) O Modelador de Processos submete as informações;
- 4) A interface aciona o Componente ProcessarModelosDeTarefa para Processar a operação de atualização no banco de dados;
- 5) O Componente cria uma conexão com banco de dados em modo transacional, cria um Objeto Recurso, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado e relaciona o objeto Papel com o ModeloDeTarefa que está sendo editado;
- 6) O componente solicita ao objeto ModeloDeTarefa para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeTarefa;
- 7) O objeto ProcessarModelosDeTarefa, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário.

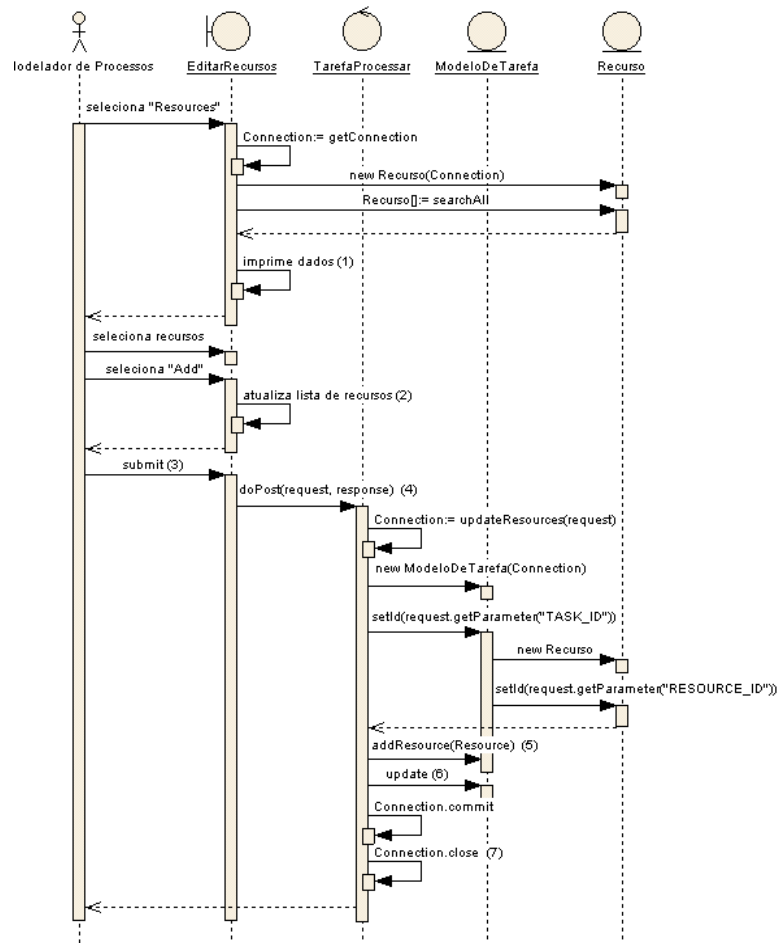


FIGURA C.8 - Diagrama de seqüência caso de uso Definir Recursos.

C.2.5.- A Seqüência: Caso de Uso Definir Tipos de Artefato de Entrada

Seqüência de mensagens dos objetos que colaboram para realização do caso de uso Definir Artefatos de Entrada mostrados na Figura C.9:

- 1) Após selecionar a opção “Inputs” na Interface de Edição de ModeloDeTarefa, é acionada a interface de Edição de artefatos de entrada, a interface aciona o objeto TipoDeArtefato que carrega os recupera os dados da base de dados, retornando o resultado que é carregado na tela;

- 2) O Modelador de Processos seleciona os tipos de artefatos disponíveis na caixa de listagem e relaciona com o ModeloDeTarefa, submetendo as informações em seguida;
- 3) O ambiente aciona o Componente ProcessarModelosDeTarefa para Processar a operação de atualização no banco de dados;
- 4) O Componente cria uma conexão com banco de dados em modo transacional, cria um objeto TipoDeArtefato, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado e relaciona o objeto Papel com o ModeloDeTarefa que está sendo editado;
- 5) O componente solicita ao objeto ModeloDeTarefa para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeTarefa;
- 6) O objeto ProcessarModelosDeTarefa, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário.

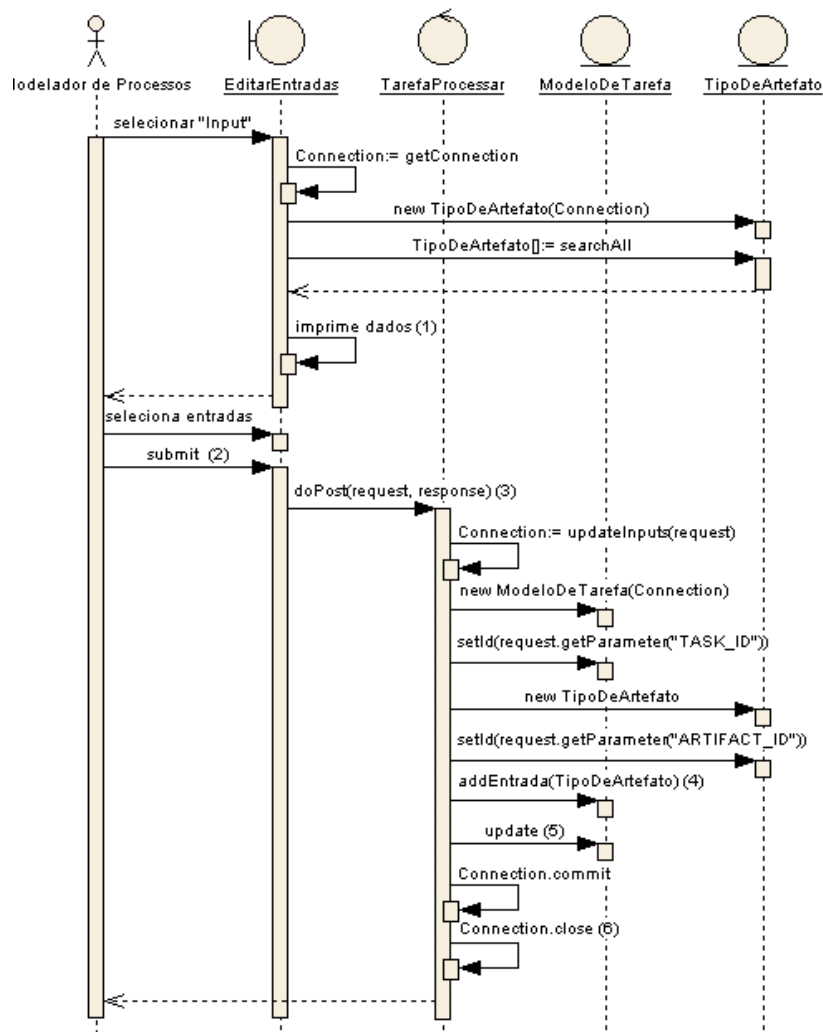


FIGURA C.9 - Diagrama de seqüência definir tipos de artefatos de entrada.

C.2.6.- A Seqüência: Caso de Uso Definir Tipos de Artefato de Saída

Seqüência de mensagens dos objetos que colaboram para realização do caso de uso Definir Artefatos de Saída mostrados na Figura C.10:

- 1) Após selecionar a opção "Outputs" na Interface de Edição de ModeloDeTarefa, é acionada a interface de Edição de artefatos de saída, a interface aciona o objeto TipoDeArtefato que carrega os recupera os dados da base de dados, retornando o resultado que é carregado na tela;

- 2) O Modelador de Processos seleciona os tipos de artefatos disponíveis na caixa de listagem e relaciona com o ModeloDeTarefa, submetendo as informações em seguida;
- 3) O ambiente aciona o Componente ProcessarModelosDeTarefa para Processar a operação de atualização no banco de dados;
- 4) O Componente cria uma conexão com banco de dados em modo transacional, cria um objeto TipoDeArtefato, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado e relaciona o objeto Papel com o ModeloDeTarefa que está sendo editado;
- 5) O componente solicita ao objeto ModeloDeTarefa para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeTarefa;
- 6) O objeto ProcessarModelosDeTarefa, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário.

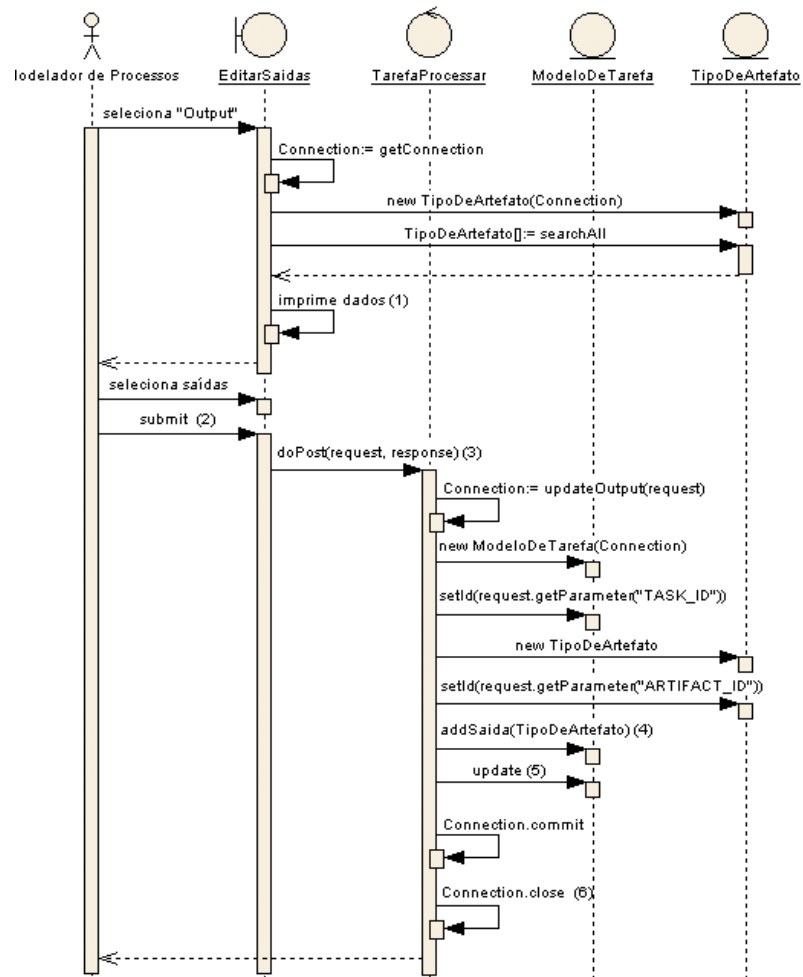


FIGURA C.10 - Diagrama de seqüência definir artefatos de saída.

C.2.7.- A Seqüência: Caso de Uso Definir Relacionamentos

Seqüência de mensagens dos objetos que colaboram para realização do cenário Definir SubTarefas do caso de uso Definir Relacionamentos mostrados na Figura C.11:

- 1) Após selecionar a opção "Subtasks" na Interface de Edição de ModeloDeTarefa, é acionada a interface de Edição de subtarefas, a interface aciona o objeto ModeloDeTarefa que recupera os dados da base de dados, retornando o resultado que é carregado na tela;

- 2) O Modelador de Processos seleciona as tarefas disponíveis na caixa de listagem e relaciona com o ModeloDeTarefa, submetendo as informações em seguida;
- 3) O ambiente aciona o Componente ProcessarModelosDeTarefa para Processar a operação de atualização no banco de dados;
- 4) O Componente cria uma conexão com banco de dados em modo transacional, cria um objeto ModeloDeTarefa, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado e relaciona os objetos Relacionamento com as sub tarefas e o ModeloDeTarefa que está sendo editado;
- 5) O componente solicita ao objeto ModeloDeTarefa para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeTarefa;
- 6) O objeto ProcessarModelosDeTarefa, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário.

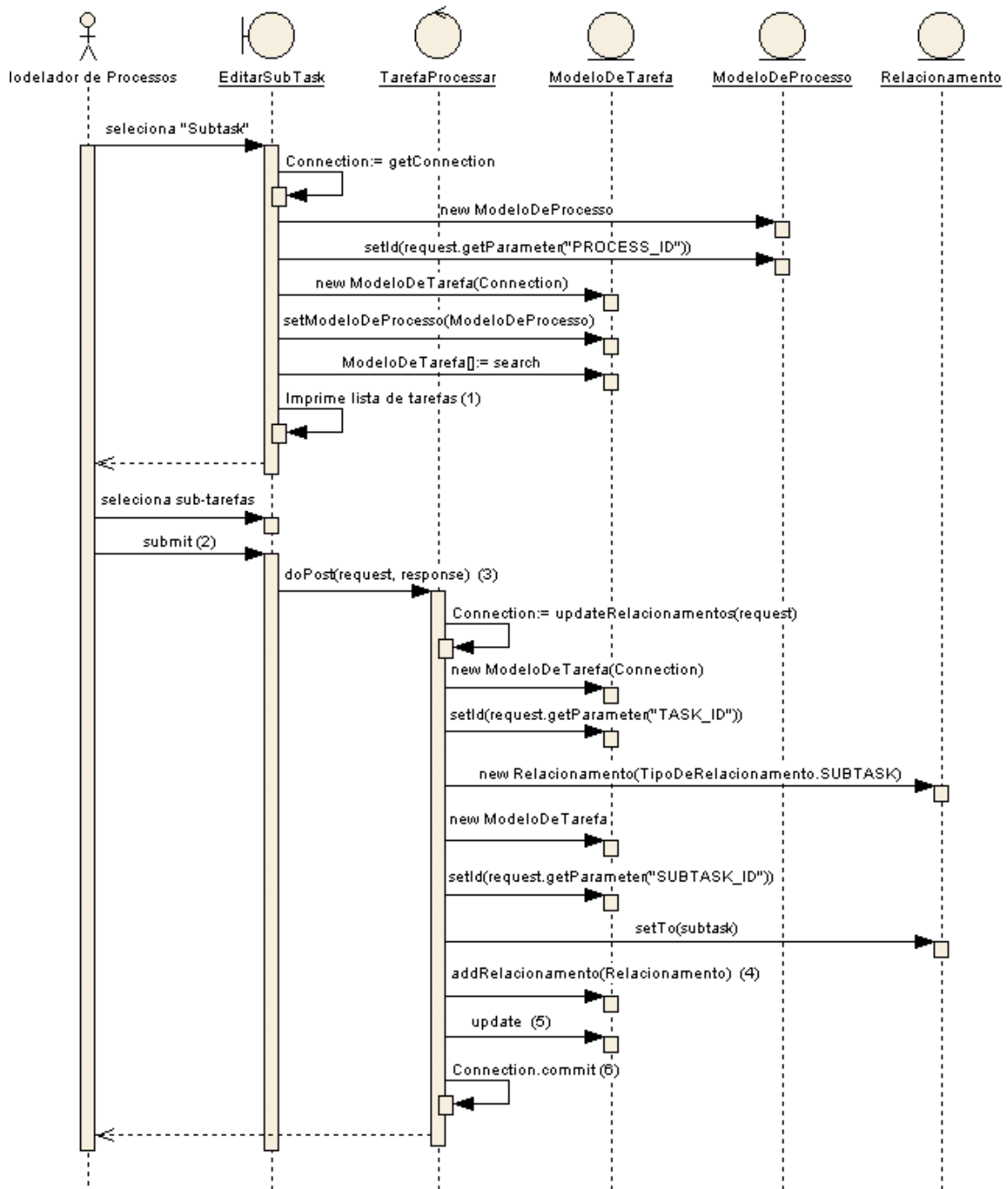


FIGURA C.11 - Diagrama de seqüência: cenário definir sub tarefas.

Seqüência de mensagens dos objetos que colaboram para realização do cenário Definir predecessoras do caso de uso Definir Relacionamentos mostrados na Figura C.12:

- 1) Após selecionar a opção “Predecessor” na Interface de Edição de ModeloDeTarefa, é acionada a interface de Edição de Predecessoras, a interface aciona os objetos ModeloDeTarefa e Relacionamento (este último retorna seus tipos) que recuperam os dados da base de dados, retornando o resultado que é carregado na tela;
- 2) O Modelador de Processos seleciona as tarefas e tipos de relacionamentos disponíveis nas caixas de listagem e relaciona com o ModeloDeTarefa clicando em “Add”, a interface atualiza a lista de recursos relacionados com a tarefa;
- 3) O Modelador de Processos submete as informações;
- 4) A interface aciona o Componente ProcessarModelosDeTarefa para Processar a operação de atualização no banco de dados;
- 5) O Componente cria uma conexão com banco de dados em modo transacional, cria um objeto ModeloDeTarefa, recupera os parâmetros enviados pela Interface de Usuário e atribui ao objeto criado e relaciona os objetos Relacionamento com as predecessoras e o ModeloDeTarefa que está sendo editado;
- 6) O componente solicita ao objeto ModeloDeTarefa para efetuar a operação no banco de dados, o componente efetua a operação retornando o resultado ao objeto ProcessarModelosDeTarefa;
- 7) O objeto ProcessarModelosDeTarefa, valida a operação, finaliza a conexão e retorna mensagem de sucesso ao usuário.

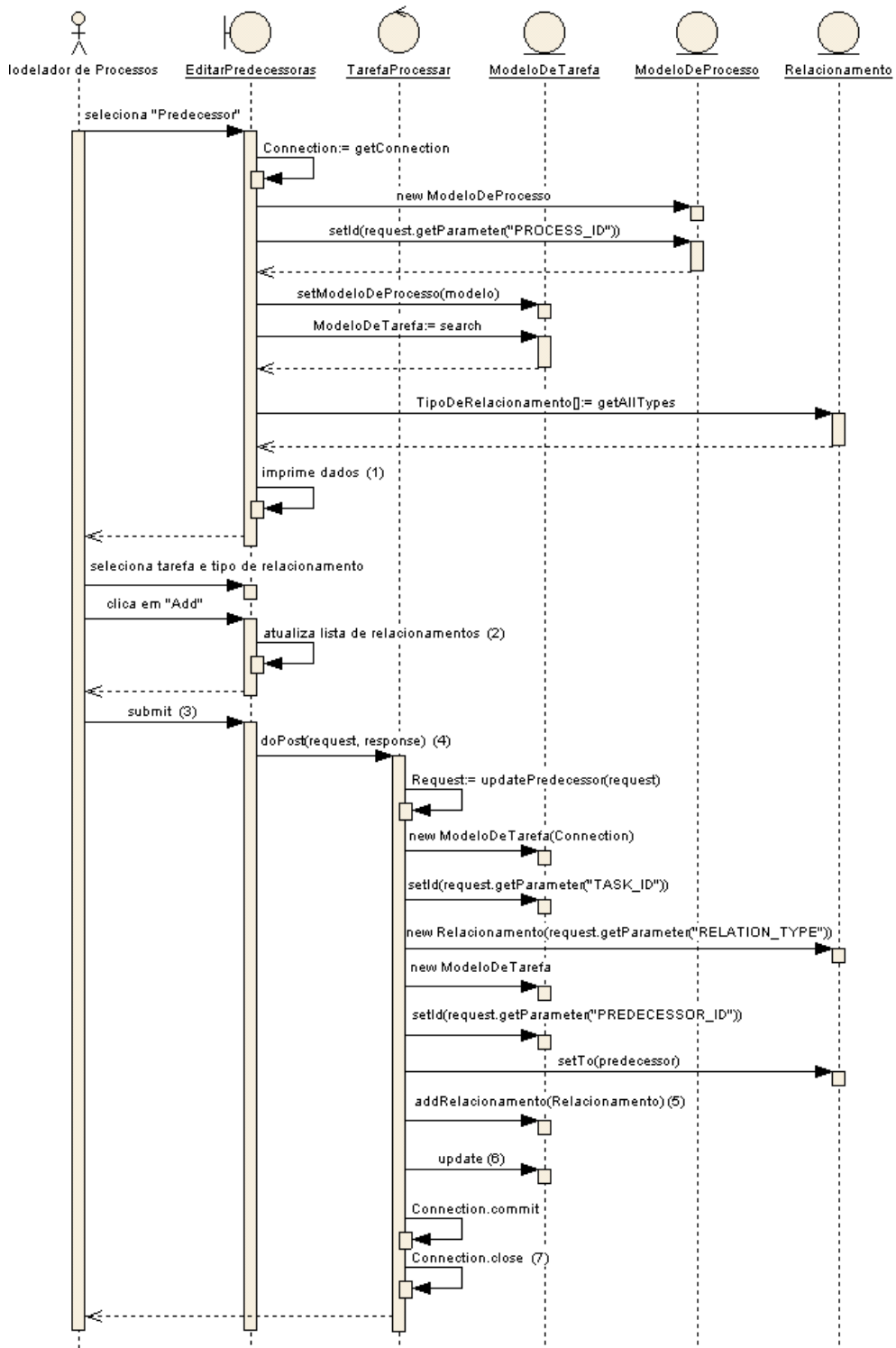


FIGURA C.12 - Diagrama de seqüência: cenário definir predecessoras.

APÊNDICE D

DESCRIÇÃO DOS CASOS DE USO DA MÁQUINA DE PROCESSOS

Neste apêndice, apresenta-se as descrições dos cenários dos casos de uso definidos para a máquina de processos conforme apresentado no Capítulo 6.

D.1.- Descrição dos Casos de Uso

Segue a descrição de cada caso de uso, para fins de identificação são apresentadas as siglas para cada sub-módulo da máquina de processos:

MCM: Sigla para Ferramenta de Configuração e Monitoração de Instâncias;

MWS: Sigla para Espaço de Trabalho Eletrônico *Workspace*.

MAS: Sigla para Serviço de Agentes

D.1.1.- Caso de Uso: Monitorar e Configurar Instâncias (Configurar execução de tarefas).

Identificador do Caso de Uso	UC-MCM-01
Nome do Caso de Uso	Monitorar e Configurar Instâncias (Configurar execução de tarefas).
Atores	Gerente de Projeto
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Gerente de Projeto deve ter sido validado pelo sistema
Requisitos Especiais	Não há
Primeiro Cenário – Configurar Instância de Processo	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none">1.- Este caso de uso inicia-se quando o gerente de projetos aciona a interface para a configuração de instâncias de processos;2.- O sistema apresentará uma lista com todos os modelos de processos configurados;3.- O Gerente de Projetos seleciona um processo;4.- O sistema apresenta um formulário oferecendo a opção de configuração automática ou manual listando todas as atividades do processo, papéis responsáveis e pessoas que podem assumir estes papéis;5.- O gerente de projeto seleciona a opção de configuração desejada, caso seja em formato manual, seleciona também as pessoas que executarão cada tarefa, exercendo um determinado papel;6.- O gerente de projetos submete as informações;7.- O sistema agrega a informações em uma mensagem e encaminha para o	

serviço de agentes.
Fluxos Alternativos No passo 6, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário
Pós Condição: O sistema apresentará mensagem de sucesso da operação.

D.1.2.- Caso de Uso: Monitorar e Configurar Instâncias (Monitorar execução de tarefas)

Identificador do Caso de Uso	UC-MCM-02
Nome do Caso de Uso	Monitorar e Configurar Instâncias (Configurar execução de tarefas)
Atores	Gerente de Projeto
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Gerente de Projeto deve ter sido validado pelo sistema. Um projeto deve ter sido selecionado pelo gerente na autenticação do usuário
Requisitos Especiais	Não há
<i>Primeiro Cenário – Monitorar e Configurar Instâncias (Configurar execução de tarefas)</i>	
Fluxo de Eventos:	
Fluxo Principal 1.- Este caso de uso inicia-se quando o gerente de projetos aciona a interface para monitoração de instâncias de processos; 2.- O sistema apresentará uma lista com todas as instâncias de processos em andamento em um dado projeto; 3.- O Gerente de Projetos seleciona uma instância; 4.- O sistema apresenta uma interface gráfica, apresentando as tarefas concluídas e em andamento, além de outras informações referentes a tarefa como tempo de execução, esforço e responsáveis e situação de cada uma formulário oferecendo a opção de configuração automática ou manual listando todas as atividades do processo, papéis responsáveis.	
Fluxos Alternativos	
Pós Condição:	

D.1.3.- Caso de Uso: Gerenciar Tarefas

Identificador do Caso de Uso	UC-MWS-01
Nome do Caso de Uso	Gerenciar Tarefas.
Atores	Desenvolvedor
Prioridade	Muito Alta
Pré Condição	Um usuário deve ter sido validado pelo sistema.
Requisitos Especiais	Não há
<i>Primeiro Cenário – Gerenciar Tarefas</i>	
Fluxo de Eventos:	
Fluxo Principal	

<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o desenvolvedor, aciona a opção “Tasks” na interface principal do e-WebProject; 2.- O sistema apresentará uma lista com todas as tarefas alocadas para o desenvolvedor; 3.- O desenvolvedor seleciona uma tarefa; 4.- O apresenta um conjunto de opções no qual o usuário pode: visualizar instruções e apontar tempos; 5.- O desenvolvedor aponta os tempos de conclusão da tarefa e solicita a conclusão da tarefa, selecionando o estado da tarefa em 100%; 6.- O desenvolvedor submete as informações; 7.- O sistema agrega a informações em uma mensagem e encaminha para o serviço de agentes.
<p>Fluxos Alternativos No passo 6, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário</p>
<p>Pós Condição: O sistema atualizará a lista de apontamentos de tempo da tarefa.</p>

D.1.4.- Caso de Uso: Controlar Instâncias de Processos (Criar Instâncias)

Identificador do Caso de Uso	UC-MAS-01
Nome do Caso de Uso	Controlar Instâncias de Processos (Criar Instâncias).
Atores	Agente Instanciador
Prioridade	Muito Alta
Pré Condição	O caso de uso UC-MCM-01 deve ter sido executado previamente.
Requisitos Especiais	Não há
<i>Primeiro Cenário – Criar Instâncias</i>	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o agente instanciador recebe uma mensagem para criação de uma nova instância de processo; 2.- O agente instanciador cria as instâncias das tarefas do processo, definindo suas condições de entrada e saída; 3.- O agente instanciador atualiza os estados das tarefas para “CRIADO” se a opção de configuração for automática ou “EM_ESPERA”, caso a configuração da instância seja planejada manual; 4.- O agente instanciador atualiza o estado da instância do processo para “ATIVO”. 5.- O agentes instanciador encaminha uma mensagem para o agente coordenador solicitando a ativação das tarefas disponíveis; 	
Fluxos Alternativos	
Caso ocorra algum problema, o estado do processo é atualizado para “ABORTADO”.	
Pós Condição:	
Não há	

D.1.5.- Caso de Uso: Controlar Instâncias de Processos (Coordenar execução)

Identificador do Caso de Uso	UC-MAS-02
Nome do Caso de Uso	Controlar Instâncias de Processos (Coordenar execução).
Atores	Agente Coordenador
Prioridade	Muito Alta
Pré Condição	Não há
Requisitos Especiais	Não há
Primeiro Cenário – Coordenar execução Tarefas	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o agente coordenador recebe uma mensagem para ativação de tarefas de processo; 2.- O agente coordenador avalia as pré condições satisfeitas das tarefas inativas; 3.- O agente coordenador envia uma mensagem para o agente mensageiro solicitando notificação de agentes humanos; 4.- O agente coordenador recebe uma mensagem do agente mensageiro e verifica se o estado da tarefa está "PRONTO" 5.- O agente coordenador atualiza os estados das tarefas para "ATIVO", e deposita na lista de tarefas do agente humano. 	
Fluxos Alternativos	
<p>No item 2, se não houverem agentes humanos planejados para a execução das tarefas, o agente coordenador envia uma mensagem para o agente planejador solicitando a melhor opção dentre os agentes humanos disponíveis para exercer os papéis da tarefa</p>	
Pós Condição:	
Não há	

D.1.6.- Caso de Uso: Controlar Instâncias de Processos (Planejar Tarefa)

Identificador do Caso de Uso	UC-MAS-03
Nome do Caso de Uso	Controlar Instâncias de Processos (Planejar Tarefa).
Atores	Agente Planejador
Prioridade	Muito Alta
Pré Condição	Não há
Requisitos Especiais	Não há
Primeiro Cenário – Planejar Tarefas	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o agente planejador recebe uma mensagem para planejar de tarefas de processo; 2.- O agente agente planejador verifica quais os papéis responsáveis pela execução do modelo de tarefa; 3.- Para cada papel, o agente planejador localiza um agente humano capaz de exercer aquele papel; 4.- O agente planejador verifica dentre os agentes humanos o que possui menos tarefa a exercer e aloca o agente humano à tarefa. 	

5.- O agente planejador atualiza os estados das tarefas para "PLANEJADO", e envia uma mensagem para o agente coordenador ativar a tarefa.
Fluxos Alternativos Não há
Pós Condição: Não há

D.1.7.- Caso de Uso: Controlar Instâncias de Processos (Notificar Agente Humano)

Identificador do Caso de Uso	UC-MAS-04
Nome do Caso de Uso	Controlar Instâncias de Processos (Notificar Agente Humano).
Atores	Agente Mensageiro
Prioridade	Muito Alta
Pré Condição	Não há
Requisitos Especiais	Um serviço SMTP (Simple Mail Transfer Protocol), deve estar disponível na infra-estrutura organizacional.
Primeiro Cenário – Coordenar execução Tarefas	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o agente mensageiro recebe uma mensagem para notificação de agentes; 2.- O agente mensageiro captura os agentes que irão executar a tarefa; 3.- O agente mensageiro recupera a lista de endereços de e-mail dos agentes humanos; 4.- O agente mensageiro cria uma mensagem de texto informando que uma nova tarefa foi alocada para o agente humano; 5.- O agente mensageiro cria uma mensagem SMTP e adiciona o texto e a lista de endereços à mensagem; 6.- O agente mensageiro envia a mensagem utilizando o serviço SMTP disponível; 7.- O agente mensageiro atualiza o estado da tarefa para "PRONTO" 8.- O agente mensageiro envia mensagem ao agente coordenador solicitando ativação da tarefa. 	
Fluxos Alternativos Caso ocorra algum problema o agente mensageiro cria um log, informando o problema ocorrido.	
Pós Condição: Não há	

D.1.8.- Caso de Uso: Controlar Instâncias de Processos (Notificar Agente Humano)

Identificador do Caso de Uso	UC-MAS-05
Nome do Caso de Uso	Controlar Instâncias de Processos (Monitorar Tarefas).
Atores	Agente Monitor
Prioridade	Muito Alta
Pré Condição	O caso de uso UC-MWS-01 deve ter sido executado.

Requisitos Especiais	Não há.
<i>Primeiro Cenário – Coordenar execução Tarefas</i>	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o agente monitor recebe uma mensagem para conclusão de uma tarefa; 2.- O agente monitor verifica as pós condições para a conclusão da tarefa; 3.- O agente monitor atualiza o estado da tarefa para “COMPLETADO”; 4.- O agente monitor envia uma mensagem para o agente coordenador ativar as próximas tarefas; 	
Fluxos Alternativos	
<p>No item 2, caso ocorra algum problema com as pós condições o desenvolvedor é informado via interface gráfica.</p> <p>No item 3, o agente monitor verifica se ainda existem tarefas para serem executadas no processo, caso não, ele atualiza o estado do processo para “COMPLETADO”.</p>	
Pós Condição:	
A lista de tarefas do desenvolvedor é atualizada mostrando a tarefa completada.	

APÊNDICE E

DESCRIÇÃO DOS CASOS DE USO DO REPOSITÓRIO DE ARTEFATOS

Apresenta-se neste apêndice, um conjunto completo da descrição dos casos de uso do repositório de artefatos.

E.1.- Descrição dos Casos de Uso

Segue a descrição de cada caso de uso, para fins de identificação é apresentada a sigla para o sub-módulo do serviço de coordenação de processos: o repositório de artefatos:

ARP: Sigla que identifica as funcionalidades do Repositório de Artefatos;

E.1.1.- Caso de Uso: Manter Tipos de Artefatos.

Identificador do Caso de Uso	UC-ARP-01
Nome do Caso de Uso	Manter Tipos de Artefatos
Atores	Gerente de Configuração
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Gerente de Configurações deve ter sido validado pelo sistema
Requisitos Especiais	Não há
<i>Primeiro Cenário – Criar Novo Tipo de Artefato</i>	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none">1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface para manter tipos de artefatos;2.- O sistema apresentará uma lista com todos os tipos de artefatos disponíveis apresentando a opção “INSERT”;3.- O Gerente de Configuração seleciona a opção “INSERT”;4.- O sistema apresenta um formulário com os dados referentes ao tipo de artefato, como nome, descrição e observação;5.- O gerente de configurações entra com os dados;6.- O gerente de configurações submete as informações;7.- O sistema verifica as informações e salva na base de dados.	
Fluxos Alternativos	
No passo 6, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário	
Pós Condição:	
O sistema apresentará mensagem de sucesso da operação.	
<i>Segundo Cenário – Editar Tipo de Artefato</i>	
Fluxo de Eventos:	

Fluxo Principal	<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface para manter tipos de artefatos; 2.- O sistema apresentará uma lista com todos os tipos de artefatos disponíveis apresentando a opção “EDIT”; 3.- O Gerente de Configuração seleciona a opção “EDIT”; 4.- O sistema apresenta um formulário com os dados atuais do tipo de artefato, como nome, descrição e observação; 5.- O gerente de configurações atualiza os dados; 6.- O gerente de gerente de configurações submete as informações; 7.- O sistema verifica as informações e salva na base de dados.
Fluxos Alternativos	No passo 6, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário
Pós Condição:	O sistema apresentará mensagem de sucesso da operação.
Terceiro Cenário – Excluir Tipo de Artefato	
Fluxo de Eventos:	
Fluxo Principal	<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface para manter tipos de artefatos; 2.- O sistema apresentará uma lista com todos os tipos de artefatos disponíveis oferecendo caixas de seleção para exclusão do tipo”; 3.- O Gerente de Configuração seleciona os tipos que deseja excluir; 4.- O Gerente de Configuração seleciona a opção “DELETE” submetendo os dados; 5.- O sistema verifica as informações e exclui da base de dados.
Fluxos Alternativos	No passo 5, se ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário
Pós Condição:	O sistema apresentará mensagem de sucesso da operação.

E.1.2.- Caso de Uso: Manter Artefatos

Identificador do Caso de Uso	UC-ARP-02
Nome do Caso de Uso	Manter Artefatos
Atores	Gerente de Configuração
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Gerente de Configurações deve ter sido validado pelo sistema O caso de uso UC-ARP-01 deve ter sido executado previamente
Requisitos Especiais	Não há
Primeiro Cenário – Criar Novo Artefato	
Fluxo de Eventos:	
Fluxo Principal	<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface de navegação de artefatos; 2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo; 3.- O gerente de configurações seleciona um tipo de artefato;

<p>4.- O sistema apresenta uma tela do lado direito informando os dados do tipo de artefato e oferecendo uma opção de criar um artefato baseado naquele tipo;</p> <p>5.- O Gerente de Configuração seleciona a opção apresentada;</p> <p>6.- O sistema apresenta um formulário com os dados referentes ao artefato, como nome, data de criação, descrição e observação;</p> <p>7.- O gerente de configurações entra com os dados;</p> <p>8.- O gerente de configurações submete as informações;</p> <p>9.- O sistema verifica as informações e salva na base de dados.</p>
<p>Fluxos Alternativos No passo 9, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário</p>
<p>Pós Condição: O sistema apresentará mensagem de sucesso da operação.</p>
<p>Segundo Cenário – Editar Artefato</p>
<p>Fluxo de Eventos:</p>
<p>Fluxo Principal</p> <p>1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface de navegação de artefatos;</p> <p>2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo;</p> <p>3.- O Gerente de Configuração expande a árvore de artefatos e seleciona o artefato que deseja editar as informações;</p> <p>4.- O sistema apresenta uma tela com as informações do artefato, apresentando algumas opções, dentre elas a de editar os dados do artefato;</p> <p>5.- O gerente de configurações seleciona a opção “Edit”;</p> <p>6.- O sistema apresenta um formulário com os dados atuais do artefato;</p> <p>7.- O gerente de configurações atualiza as informações do artefato;</p> <p>8.- O gerente de configurações submete as informações;</p> <p>9.- O sistema verifica as informações e salva na base de dados.</p>
<p>Fluxos Alternativos No passo 9, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário</p>
<p>Pós Condição: O sistema apresentará mensagem de sucesso da operação.</p>
<p>Terceiro Cenário – Excluir Artefato</p>
<p>Fluxo de Eventos:</p>
<p>Fluxo Principal</p> <p>1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface de navegação de artefatos;</p> <p>2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo;</p> <p>3.- O Gerente de Configuração seleciona expande a árvore de artefatos e seleciona o artefato que deseja editar as informações;</p> <p>4.- O sistema apresenta uma tela com as informações do artefato, apresentando algumas opções, dentre elas a de excluir o artefato;</p> <p>5.- O gerente de configurações seleciona a opção “Delete”;</p> <p>6.- O sistema apresenta um formulário de confirmação de exclusão;</p> <p>7.- O gerente de configurações confirma a exclusão e submete as informações;</p> <p>8.- O sistema verifica as informações e exclui da base de dados.</p>
<p>Fluxos Alternativos No passo 8, se ocorrer alguma operação ilegal, o sistema apresentará mensagem</p>

de erro ao usuário
Pós Condição: O sistema apresentará mensagem de sucesso da operação.

E.1.3.- Caso de Uso: Manter Versões

Identificador do Caso de Uso	UC-ARP-03
Nome do Caso de Uso	Manter Versões
Atores	Gerente de Configuração
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Gerente de Configurações deve ter sido validado pelo sistema O caso de uso UC-ARP-02 deve ter sido executado previamente
Requisitos Especiais	Não há

Primeiro Cenário – Criar Nova Versão

Fluxo de Eventos:

Fluxo Principal

- 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface de navegação de artefatos;
- 2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo;
- 3.- O gerente de configurações expande a árvore e seleciona um artefato;
- 4.- O sistema apresenta uma tela do lado direito informando os dados do artefato, incluindo uma listagem de suas respectivas versões oferecendo uma opção de criar uma nova versão;
- 5.- O Gerente de Configuração seleciona a opção apresentada;
- 6.- O sistema apresenta um formulário com os dados referentes a versão, como nome, data de criação, descrição e uma opção para selecionar um arquivo que será carregado na base;
- 7.- O gerente de configurações entra com os dados e seleciona o arquivo;
- 8.- O gerente de configurações submete as informações;
- 9.- O sistema verifica as informações e salva na base de dados.

Fluxos Alternativos

No passo 9, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário

Pós Condição:

O sistema apresentará mensagem de sucesso da operação.

Segundo Cenário – Editar Versão

Fluxo de Eventos:

Fluxo Principal

- 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface de navegação de artefatos;
- 2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo;
- 3.- O gerente de configurações expande a árvore e seleciona um artefato;
- 4.- O sistema apresenta uma tela do lado direito informando os dados do artefato, incluindo uma listagem de suas respectivas versões oferecendo uma opção de editar uma versão;
- 5.- O Gerente de Configuração seleciona a opção apresentada;
- 6.- O sistema apresenta um formulário com os dados atuais da versão, como nome,

<p>data de criação, descrição e uma opção para selecionar um arquivo que será sobreposto ao arquivo que está na base;</p> <p>7.- O gerente de configurações atualiza os dados e seleciona o arquivo;</p> <p>8.- O gerente de configurações submete as informações;</p> <p>9.- O sistema verifica as informações e salva na base de dados.</p>
<p>Fluxos Alternativos</p> <p>No passo 9, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário</p>
<p>Pós Condição:</p> <p>O sistema apresentará mensagem de sucesso da operação.</p>
<p>Terceiro Cenário – Excluir Versão</p>
<p>Fluxo de Eventos:</p>
<p>Fluxo Principal</p> <p>1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface de navegação de artefatos;</p> <p>2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo;</p> <p>3.- O Gerente de Configuração seleciona expande a árvore de artefatos e seleciona o artefato que deseja editar as informações;</p> <p>4.- O sistema apresenta uma tela do lado direito informando os dados do artefato, incluindo uma listagem de suas respectivas versões oferecendo uma opção de seleção de exclusão de cada versão;</p> <p>5.- O gerente de configurações seleciona as versões que deseja excluir</p> <p>6.- O gerente de configurações seleciona a opção “Delete” submetendo as informações;</p> <p>7.- O sistema verifica as informações e exclui da base de dados.</p>
<p>Fluxos Alternativos</p> <p>No passo 7, se ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário</p>
<p>Pós Condição:</p> <p>O sistema apresentará mensagem de sucesso da operação.</p>

E.1.4.- Caso de Uso: Manter Configurações.

Identificador do Caso de Uso	UC-ARP-04
Nome do Caso de Uso	Manter Configurações
Atores	Gerente de Configuração
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Gerente de Configurações deve ter sido validado pelo sistema
Requisitos Especiais	Não há
Primeiro Cenário – Criar Nova Configuração	
Fluxo de Eventos:	
Fluxo Principal	
<p>1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface para manter configurações;</p> <p>2.- O sistema apresentará uma lista com todas as configurações de releases disponíveis apresentando a opção “INSERT”;</p> <p>3.- O Gerente de Configuração seleciona a opção “INSERT”;</p> <p>4.- O sistema apresenta um formulário com os dados referentes à configuração como nome, data de criação, versão, descrição e observação;</p> <p>5.- O gerente de configurações entra com os dados;</p>	

6.- O gerente de configurações submete as informações; 7.- O sistema verifica as informações e salva na base de dados
Fluxos Alternativos Nos passos 7 e 11, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário
Pós Condição: O sistema apresentará mensagem de sucesso da operação.
Segundo Cenário – Editar Configuração
Fluxo de Eventos:
Fluxo Principal 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface para manter configurações; 2.- O sistema apresentará uma lista com todos as configurações de releases disponíveis apresentando a opção “EDIT”; 3.- O Gerente de Configuração seleciona a opção “EDIT”; 4.- O sistema apresenta um formulário com os dados atuais da configuração, como nome, descrição e observação; 5.- O gerente de configurações atualiza os dados; 6.- O gerente de gerente de configurações submete as informações; 7.- O sistema verifica as informações e salva na base de dados.
Fluxos Alternativos No passo 6, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário
Pós Condição: O sistema apresentará mensagem de sucesso da operação.
Terceiro Cenário – Excluir Configuração
Fluxo de Eventos:
Fluxo Principal 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface para manter configurações; 2.- O sistema apresentará uma lista com todos as configurações de releases disponíveis oferecendo caixas de seleção para exclusão da configuração; 3.- O Gerente de Configuração seleciona os tipos que deseja excluir; 4.- O Gerente de Configuração seleciona a opção “DELETE” submetendo os dados; 5.- O sistema verifica as informações e exclui da base de dados.
Fluxos Alternativos No passo 5, se ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário
Pós Condição: O quarto cenário deste caso de uso deve ser executado.
Quarto Cenário – Atualizar lista de versões de artefatos
Fluxo de Eventos:
Fluxo Principal 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a opção “artifact versions” na interface de edição das configurações; 2.- O sistema aciona uma interface de edição dos dados da configuração, permitindo adicionar ou remover versões de artefatos da configuração; 3.- O gerente de configurações seleciona o artefato e respectiva versão; 4.- O gerente de configurações seleciona “Add” ou seleciona as versões quer deseja excluir e seleciona “Remove”

5.- O sistema salva as informações na base de dados.
Fluxos Alternativos No passo 5, se ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário
Pós Condição: O sistema atualizará a lista de versões e apresentará ao gerente de configurações.

E.1.5.- Caso de Uso: Congelar Versão.

Identificador do Caso de Uso	UC-ARP-05
Nome do Caso de Uso	Congelar Versão
Atores	Gerente de Configuração
Prioridade	Alta
Pré Condição	Um usuário com perfil Gerente de Configurações deve ter sido validado pelo sistema O caso de uso UC-ARP-03 deve ter sido executado previamente
Requisitos Especiais	Não há
Primeiro Cenário – Criar Nova Configuração	
Fluxo de Eventos:	
Fluxo Principal 1.- Este caso de uso inicia-se quando o gerente de configuração aciona a interface de navegação de artefatos; 2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo; 3.- O gerente de configurações expande a árvore e seleciona um artefato; 4.- O sistema apresenta uma tela do lado direito informando os dados do artefato, incluindo uma listagem de suas respectivas versões oferecendo uma opção de congelar uma versão; 5.- O Gerente de Configuração seleciona a opção apresentada submetendo os dados; 6.- O sistema verifica as informações e salva na base de dados.	
Fluxos Alternativos Nos passo 6, se o sistema verificar que ocorreu alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário	
Pós Condição: O sistema apresentará atualizará a lista de versões desabilitando as opções de check-in, check-out e edit da versão.	

E.1.6.- Caso de Uso: Criar Novo Artefato.

Identificador do Caso de Uso	UC-ARP-06
Nome do Caso de Uso	Criar Novo Artefato
Atores	Executor
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Executor deve ter sido validado pelo sistema
Requisitos Especiais	As opções de editar e excluir artefatos não devem estar disponíveis
Primeiro Cenário – Criar Novo Artefato	
Fluxo de Eventos:	
Fluxo Principal	

<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o executor aciona a interface de navegação de artefatos; 2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo; 3.- O executor seleciona um tipo de artefato; 4.- O sistema apresenta uma tela do lado direito informando os dados do tipo de artefato e oferecendo uma opção de criar um artefato baseado naquele tipo; 5.- O executor seleciona a opção apresentada; 6.- O sistema apresenta um formulário com os dados referentes ao artefato, como nome, data de criação, descrição e observação; 7.- O executor entra com os dados; 8.- O executor submete as informações; 9.- O sistema verifica as informações e salva na base de dados.
<p>Fluxos Alternativos</p> <p>No passo 9, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário</p>
<p>Pós Condição:</p> <p>O sistema apresentará mensagem de sucesso da operação.</p>

E.1.7.- Caso de Uso: Criar Nova Versão.

Identificador do Caso de Uso	UC-ARP-07
Nome do Caso de Uso	Criar Nova Versão
Atores	Executor
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Executor deve ter sido validado pelo sistema O caso de uso UC-ARP-02 deve ter sido executado previamente
Requisitos Especiais	As opções de edição e exclusão de versões devem ser desabilitadas
Primeiro Cenário – Criar Nova Versão	
Fluxo de Eventos:	
Fluxo Principal	
<ol style="list-style-type: none"> 1.- Este caso de uso inicia-se quando o executor aciona a interface de navegação de artefatos; 2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo; 3.- O executor expande a árvore e seleciona um artefato; 4.- O sistema apresenta uma tela do lado direito informando os dados do artefato, incluindo uma listagem de suas respectivas versões oferecendo uma opção de criar uma nova versão; 5.- O executor seleciona a opção apresentada; 6.- O sistema apresenta um formulário com os dados referentes a versão, como nome, data de criação, descrição e uma opção para selecionar um arquivo que será carregado na base; 7.- O executor entra com os dados e seleciona o arquivo; 8.- O executor submete as informações; 9.- O sistema verifica as informações e salva na base de dados. 	
<p>Fluxos Alternativos</p> <p>No passo 9, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário</p>	

Pós Condição: O sistema atualizará a lista de versões.

E.1.8.- Caso de Uso: Check-out de Versão.

Identificador do Caso de Uso	UC-ARP-08
Nome do Caso de Uso	Check-out de Versão
Atores	Executor
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Executor deve ter sido validado pelo sistema O caso de uso UC-ARP-03 ou UC-ARP-07 deve ter sido executado previamente
Requisitos Especiais	As opções de edição e exclusão de versões devem ser desabilitadas

Primeiro Cenário – Check-Out de Versão

Fluxo de Eventos:

Fluxo Principal

- 1.- Este caso de uso inicia-se quando o executor aciona a interface de navegação de artefatos;
- 2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo;
- 3.- O executor expande a árvore e seleciona um artefato;
- 4.- O sistema apresenta uma tela do lado direito informando os dados do artefato, incluindo uma listagem de suas respectivas versões oferecendo uma opção para check-out de versão;
- 5.- O executor seleciona a opção apresentada;
- 6.- O sistema verifica as informações e salva na base de dados.

Fluxos Alternativos

No passo 6, se o sistema verificar que ocorreu alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário

Pós Condição:

O sistema atualizará a lista de versões, desabilitará as opções de check-in e check-out para outros usuários com perfil de executor, indicará a mudança de estado da versão para "TRAVADA" mostrando o nome do usuário que efetuou a operação.

E.1.9.- Caso de Uso: Check-in de Versão.

Identificador do Caso de Uso	UC-ARP-09
Nome do Caso de Uso	Check-in de Versão
Atores	Executor
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Executor deve ter sido validado pelo sistema O caso de uso UC-ARP-08 deve ter sido executado previamente
Requisitos Especiais	As opções de edição e exclusão de versões devem ser desabilitadas

Primeiro Cenário – Check-In de Versão

Fluxo de Eventos:

Fluxo Principal

- 1.- Este caso de uso inicia-se quando o executor aciona a interface de navegação

<p>de artefatos;</p> <p>2.- O sistema apresentará uma árvore do lado esquerdo, que lista os artefatos categorizados por projeto e depois por tipo;</p> <p>3.- O executor expande a árvore e seleciona um artefato;</p> <p>4.- O sistema apresenta uma tela do lado direito informando os dados do artefato, incluindo uma listagem de suas respectivas versões oferecendo uma opção para check-in de versão;</p> <p>5.- O executor seleciona a opção apresentada;</p> <p>6.- O sistema apresenta uma tela informando os dados referentes a versão, como nome, data de criação, descrição e uma opção para selecionar um arquivo que será carregado na base;</p> <p>7.- O executor seleciona o arquivo;</p> <p>8.- O executor submete as informações;</p> <p>9.- O sistema verifica as informações e salva na base de dados.</p>
<p>Fluxos Alternativos</p> <p>No passo 9, se o sistema verificar que algum campo obrigatório não foi informado ou ocorrer alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário</p>
<p>Pós Condição:</p> <p>O sistema deve executar o caso de uso UC-ARP-10.</p>

E.1.10.- Caso de Uso: Manter Histórico de Modificações.

Identificador do Caso de Uso	UC-ARP-10
Nome do Caso de Uso	Manter Histórico de Modificações
Atores	Executor
Prioridade	Muito Alta
Pré Condição	Um usuário com perfil Executor deve ter sido validado pelo sistema O caso de uso UC-ARP-09 deve ter sido executado previamente
Requisitos Especiais	As opções de edição e exclusão de versões devem ser desabilitadas
Primeiro Cenário – Manter Histórico de Modificações	
Fluxo de Eventos:	
Fluxo Principal	
<p>1.- Este caso de uso inicia-se executa o caso de uso UC-ARP-10;</p> <p>2.- O sistema apresentará um formulário, apresentando os dados da modificação, como: data da modificação, descrição e preencherá os dados do autor automaticamente;</p> <p>3.- O executor entra com os dados</p> <p>4.- O executor submete as informações;</p> <p>5.- O sistema verifica as informações e salva na base de dados.</p>	
Fluxos Alternativos	
<p>No passo 5, se o sistema verificar que algum campo obrigatório não foi informado ou ocorreu alguma operação ilegal, o sistema apresentará mensagem de erro ao usuário</p>	
Pós Condição:	
<p>O sistema atualizará a lista de versões, abilitará as opções de check-in e check-out para outros usuários com perfil de executor, indicará a mudança de estado da versão para “LIBERADA”.</p>	